UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS RUSSAS

CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

CARLOS VICTOR DANTAS ARAUJO

UTILIZAÇÃO DE DESIGUALDADES VÁLIDAS BASEADAS EM CONDIÇÕES DE OTIMALIDADE NA CONSTRUÇÃO DE ALGORITMOS HEURÍSTICOS PARA O PROBLEMA DO CORTE MÁXIMO

RUSSAS

2018

CARLOS VICTOR DANTAS ARAUJO

UTILIZAÇÃO DE DESIGUALDADES VÁLIDAS BASEADAS EM CONDIÇÕES DE OTIMALIDADE NA CONSTRUÇÃO DE ALGORITMOS HEURÍSTICOS PARA O PROBLEMA DO CORTE MÁXIMO

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação do Campus Russas da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Ciência da Computação.

Orientador: Prof. Dr. Pablo Luiz Braga Soares

RUSSAS

2018

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

A688u Araujo, Carlos Victor Dantas.
Utilização de desigualdades válidas baseadas em condições de otimalidade na construção de algoritmos heurísticos para o Problema do Corte Máximo / Carlos Victor Dantas Araujo. – 2018.
60 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Russas, Curso de Ciência da Computação, Russas, 2018.
Orientação: Prof. Dr. Pablo Luiz Braga Soares.

1. Problema do Corte Máximo. 2. Desigualdades Válidas. 3. Heurísticas. 4. Algoritmo Genético. 5. Têmpera Simulada. I. Título.

CDD 005

CARLOS VICTOR DANTAS ARAUJO

UTILIZAÇÃO DE DESIGUALDADES VÁLIDAS BASEADAS EM CONDIÇÕES DE OTIMALIDADE NA CONSTRUÇÃO DE ALGORITMOS HEURÍSTICOS PARA O PROBLEMA DO CORTE MÁXIMO

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação do Campus Russas da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Ciência da Computação.

Aprovada em:

BANCA EXAMINADORA

Prof. Dr. Pablo Luiz Braga Soares (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Ms. Tatiane Fernandes Figueiredo
Universidade Federal do Ceará (UFC)

Prof. Dr. Bonfim Amaro Júnior
Universidade Federal do Ceará (UFC)

Dedico este trabalho a toda minha família, em especial ao meu pai Carlos Gonzaga Napolião Araújo e minha mãe Veralúcia Dantas de Holanda.

AGRADECIMENTOS

Ao Prof. Dr. Pablo Luiz Braga Soares por além de me orientar nesse trabalho de conclusão de curso, ser meu amigo e ter me ajudado em diversos momentos.

A Prof.ª Tatiane Fernandes Figueiredo por me orientar em outras pesquisas e me ensinar tanto.

Aos meus amigos da Universidade Federal do Ceará, em especial Afonso Matheus, Alex Frederico, Elis Vieira, Erik Almeida, Hugo Venâncio, Igor Mendes, Isaac Rahel, Isaias Ferreira, Marcos Alencar, Marcos Paulo, Marília Cristina, Paloma Bispo, Pedro Jofre, Sabrina Oliveira, Tágila Lima, Thomas Dillan e Vinicius Almeida e Yan Vancelis, que estiveram comigo em tantos momentos e dividiram essa árdua batalha da graduação.

Aos amigos que dividiram teto comigo, Daniel Lira, João Pedro, Mateus Oliveira e Matheus Carneiro, foi um prazer compartilhar esse tempo com vocês.

A todos os integrantes do núcleo de estudos do NEMO, que nas diversas horas de dedicação à pesquisa tornaram o trabalho mais prazeroso e divertido.

As amizades que estão comigo desde antes da universidade, em especial Alcione Maia, Aluísio Nascimento, Bruno Chaves, Camila Sabino, Fabinha Silva, Ítalo Neves, Ivo Nogueira, Joedna Flor, Marcos Araújo, Vanessa Scomparim, Victor Maia e Yasmim Acioli, que a muito tempo contribuem na minha vida e nas minhas vitórias.

Aos meus pais, irmãos, tios e avós, que nos momentos de minha ausência dedicados ao estudo, sempre fizeram entender que o futuro é feito a partir da dedicação no presente!

Agradeço a todos os professores por me proporcionar o conhecimento não apenas racional, mas a manifestação do caráter e afetividade da educação no processo de formação profissional, por tanto que se dedicaram a mim, não somente por terem me ensinado, mas por terem me feito aprender.

Ao Doutorando em Engenharia Elétrica, Ednardo Moreira Rodrigues, e seu assistente, Alan Batista de Oliveira, aluno de graduação em Engenharia Elétrica, pela adequação do *template* utilizado neste trabalho para que o mesmo ficasse de acordo com as normas da biblioteca da Universidade Federal do Ceará (UFC).

”Aonde fica a saída?”, Perguntou Alice ao gato que ria.

”Depende”, respondeu o gato.

”De quê?”, replicou Alice;

”Depende de para onde você quer ir...”

(Alice no País das Maravilhas; 1951)

RESUMO

O Problema do Corte Máximo (*Max-Cut*) consiste em particionar um conjunto de n vértices em dois subconjuntos de modo que a soma das arestas entre os subconjuntos seja a maior possível. Esse trabalho de conclusão de curso apresenta 4 novas versões de algoritmos heurísticos, baseadas em Algoritmo Genético e Têmpera Simulada, que utilizam um conjunto de desigualdades válidas em sua estrutura. No melhor do nosso conhecimento, somos os primeiros a utilizar desigualdades válidas na construção de heurísticas para o Problema do Corte Máximo. Os resultados dos experimentos computacionais mostram que o uso dessas desigualdades faz com que os algoritmos obtenham melhores resultados que a versão clássica, ou seja, sem as desigualdades e também sejam, em alguns casos, competitivos com o estado da arte para o *Max-Cut*.

Palavras-chave: Problema do Corte Máximo. Desigualdades Válidas. heurísticas. Algoritmo Genético. Têmpera Simulada.

ABSTRACT

The Max-Cut problem consists of partitioning a set of n nodes into two subsets so that the sum of the edges between the subsets is large as possible. This Course Completion Paper presents 4 new versions of heuristics algorithms, based on Genetic Algorithm and Simulated Annealing, which use a set of valid inequalities in their structure. To the best of our knowledge, we are the first to use valid inequalities in the construction of metaheuristics for the Max-Cut Problem. The results of the computational experiments show that the use of these inequalities causes the algorithms to obtain better results than the classical version, that is, without the inequalities and also, in some cases, competitive with the state of the art for *Max Cut*.

Keywords: Max-Cut Problem. Valid inequalities. heuristics. Genetic Algorithm. Simulated Annealing

LISTA DE FIGURAS

Figura 1 – (a) O corte que maximiza o valor de soma das arestas que estão entre os subconjuntos S (Nós azuis) e $\bar{S}$ (Nós brancos). 25
Figura 2 – Representação do comportamento das desigualdades válidas para a solução ótima da Figura 1. 34
Figura 3 – Gráfico de Comparação da variabilidade genética da instância *G4*. 42
Figura 4 – Gráfico de Comparação da variabilidade genética da instância *G45*. 43
Figura 5 – Gráfico de Comparação da variabilidade genética da instância *G24*. 43
Figura 6 – Gráfico de Comparação da variabilidade genética da instância *G50*. 44

LISTA DE TABELAS

Tabela 1 – Comparação dos GA's para instâncias pertencentes ao conjuntos *G* 45
Tabela 2 – Compração dos GA's para instâncias pertencentes ao conjuntos *statistical_physics_application* 46
Tabela 3 – Comparação dos algoritmos *AG-HF, IGA* e *GA-VNS* utilizando as instâncias do conjunto *G* 48
Tabela 4 – Comparação dos SA's para as instâncias pertencentes ao conjunto *G* 51
Tabela 5 – Comparação dos SA's para as instâncias pertencentes ao conjunto *statistical_physics_application* 52
Tabela 6 – Comparação dos algoritmos SA-H e AG-HF para as instâncias pertencentes ao conjunto *G* 54
Tabela 7 – Comparação dos algoritmos SA-H e AG-HF para as instâncias pertencentes ao conjunto *statistical_physics_application* 55

LISTA DE ALGORITMOS

Algoritmo 1 – Algoritmo Genético Clássico 22
Algoritmo 2 – Têmpera Simulada 24

Algoritmo 3 – Criação de indivíduos utilizando as Desigualdades 35
Algoritmo 4 – GA-PI 36
Algoritmo 5 – GA-M 37
Algoritmo 6 – GA - HF 38
Algoritmo 7 – SA-H 39

LISTA DE ABREVIATURAS E SIGLAS

Max-Cut	Problema do Corte Máximo
AG	Algoritmo Genético
AGC	Algoritmo Genético Clássico
AGI	Algoritmo Genético Incremental
FO	Função Objetivo
GRASP	*Greedy Randomized Adaptive Search Procedure*
PLI	Programação Linear Inteira
PO	Pesquisa Operacional
QUBO	Otimização Quadrática Binária sem Restrições
SA	*Simulated Annealing*
VND	*Variable Neighborhood Descent*
VNS	*Variable Neighborhood Search*

SUMÁRIO

1 INTRODUÇÃO 15
1.1 Objetivos 16
1.1.1 Objetivos Gerais 16
1.1.2 Objetivos Específicos 16
1.2 Metodologia 16
1.3 Organização do Trabalho 17
2 REFERENCIAL TEÓRICO 18
2.1 Conceitos Necessários de Teoria dos Grafos 18
2.2 Conceitos Necessários de Pesquisa Operacional 18
2.2.1 Programação Linear 19
2.2.2 Programação Linear Inteira 20
2.2.3 Desigualdades Válidas 20
2.2.4 Métodos Heurísticos 21
2.2.4.1 Algoritmo Genético *21*
2.2.4.2 Têmpera Simulada *23*
3 PROBLEMA DO CORTE MÁXIMO 25
3.1 Max-Cut 25
3.2 Formulações 26
3.2.1 Formulação Binária {0, 1} 26
3.2.2 Formulação Binária {-1, 1} 26
3.2.3 Demais Formulações 26
3.3 Condições de Otimalidade 27
4 APLICAÇÕES 29
4.1 Spin Glasses 29
4.2 VLSI 30
4.3 Segmentação de Imagens 30
5 REVISÃO DA LITERATURA 32
6 ALGORITMOS PROPOSTOS 34
6.1 Codificação das Desigualdades Válidas 34

6.2 **Algoritmo Genético - Desigualdades Válidas na Geração da População Inicial** 35
6.3 **Algoritmo Genético - Desigualdades Válidas Como Função de Mutação** 36
6.4 **Algoritmo Genético - Desigualdades Válidas em Ambas as Etapas** 37
6.5 **Têmpera Simulada - Desigualdades Válidas como Modificador da Solução Corrente** 38
7 **RESULTADOS COMPUTACIONAIS** 40
7.1 **Configuração do Ambiente Computacional** 40
7.2 **Instancias** 40
7.2.1 ***Características das Instâncias*** 40
7.3 **Algoritmos Baseados em Algoritmo Genético** 41
7.4 **Algoritmo Baseado em Têmpera Simulada** 49
7.5 **Algoritmo Genético e Têmpera Simulada** 53
8 **CONCLUSÃO** 56
8.1 **Trabalhos Futuros** 56
REFERÊNCIAS 58

1 INTRODUÇÃO

O Problema do Corte Máximo (*Max-Cut*) é um problema da área de otimização combinatória que, devido ao seu vasto domínio de aplicações, tais como física estatística, segmentação de imagens, circuitos integrados de escala muito grande (*Very Large Scale Integrated* (VLSI) Circuits) etc (ANJOS, 2001; BARAHONA *et al.*, 1988; SOUSA *et al.*, 2013; LIERS *et al.*, 2011; BOYKOV; JOLLY, 2001), tem atraído o interesse científico de diversos pesquisadores nas áreas de otimização matemática e matemática discreta (HELMBERG *et al.*, 1996; BURER *et al.*, 2001/2002; DEZA; LAURENT, 1997; STEVEN *et al.*, 1998). A quantidade de problemas práticos e matemáticos que aparecem diretamente na forma do *Max-Cut* ou que podem ser modelados de tal maneira, faz com que cada vez mais seja necessário o desenvolvimento de algoritmos que obtenham bons resultados em tempo viável.

Existem, no entanto, algumas desvantagens na utilização de algoritmos exatos para o *Max-Cut*. Por ser um problema NP-Difícil (KARP, 1972), é considerado um desafio computacional resolvê-lo de forma ótima, mesmo para instâncias de tamanhos moderados, cerca de 200 vértices (KRISLOCK *et al.*, 2014). Por conseguinte, para instâncias desse tamanho ou maiores, faz-se necessário o uso de abordagens que utilizem técnicas de *branch-and-cut, branch-and-price*, desigualdades válidas ou algoritmos que forneçam soluções próximas da solução ótima, tais como heurísticas, meta-heurísticas e algoritmos aproximativos (DUNNING *et al.*, 2018; GOEMANS; WILLIAMSON, 1995a).

No melhor do nosso conhecimento, somos os primeiros a construir algoritmos heurísticos que utilizam, em sua estrutura, um conjunto de desigualdades válidas para o *Max-Cut*, existindo na literatura apenas a utilização de desigualdades válidas ou desigualdades válidas em técnicas exatas de resolução (SIMONE *et al.*, 1995; SIMONE; RINALDI, 1994; RENDL *et al.*, 2007). Mais especificamente, somos os primeiros a utilizar o conjunto de desigualdades válidas provado em (SOARES, 2018). Neste trabalho, são apresentadas resultados de 6 diferentes heurísticas, juntamente com uma análise dos resultados obtidos através de extensos experimentos computacionais realizados em instâncias da literatura. Os resultados da análise indicam que a utilização do conjunto de desigualdades válidas faz com que as heurísticas obtenham melhores resultados, tornando-os competitivos com trabalhos existentes e em alguns casos, conseguindo se igualar ao estado da arte para o *Max-Cut*.

1.1 Objetivos

1.1.1 Objetivos Gerais

Desenvolver algoritmos heurísticos aplicando, em sua estrutura, um conjunto de desigualdades válidas. Os algoritmos apresentados neste trabalho serão baseados em duas meta-heurísticas: Algoritmo Genético (AG) e Têmpera Simulada (*Simulated Annealing* (SA)).

1.1.2 Objetivos Específicos

- Revisar formulações matemáticas para o problema proposto;
- Apresentar a prova matemática, presente em (SOARES, 2018), das condições de otimalidade;
- Implementar diferentes versões das heurísticas baseadas nas meta-heurísticas selecionadas;
- Efetuar uma análise de eficiência das diferentes implementações;
- Comparar as versões com trabalhos da literatura.

1.2 Metodologia

Dada a natureza aplicada da pesquisa realizada, o trabalho será desenvolvido a partir de estudos aprofundados da área de Teoria do Grafos, Pesquisa Operacional (PO) e algumas das formulações matemáticas existentes para o *Max-Cut*. Será utilizado um método de prova para demonstrar a certeza matemática a respeito do conjunto de desigualdades válidas utilizado.

Neste trabalho serão implementadas 6 algoritmos, onde 4 deles serão algoritmos heurísticos que utilizarão o desigualdades válidas e os outros 2 não. Dos 6 algoritmos, 4 são baseadas em AG e serão implementadas no seguinte padrão:

1. O primeiro algoritmo é um Algoritmo Genético Clássico (AGC) (ver Seção 2.2.4.1), ou seja, não utiliza as desigualdades;
2. O segundo algoritmo utilizará as desigualdades na etapa de geração da população inicial;
3. O terceiro algoritmo utilizará as desigualdades como função de mutação da população;
4. O quarto, e último algoritmo baseado em AG, aplica as restrições em ambas as etapas citadas nos algoritmos 2 e 3.

Os dois últimos algoritmos implementadas serão baseados em SA, uma versão sem o desigualdades válidas e outra utilizando as desigualdades como otimizador da solução corrente. Em

seguida, os algoritmos apresentados serão alimentados com instâncias da literatura geradas por (HELMBERG; RENDL, 2000; LIERS *et al.*, 2004; LIERS, 2004), a fim de comparar seus desempenhos. A análise dos algoritmos heurísticos apresentados levará em consideração fatores como o tempo de execução e valor obtido na Função Objetivo (FO).

1.3 Organização do Trabalho

Este trabalho se divide em 8 capítulos. O Capítulo 1 introduz de forma breve a problemática e a importância do problema abordado, assim como os objetivos esperados no decorrer da pesquisa e a metodologia utilizada. O Capítulo 2 apresenta os conceitos base necessários para entendimento e resolução do problema proposto. O Capítulo 3 define matematicamente o *Max-Cut* e apresenta o conjunto desigualdades válidas que será utilizado neste trabalho de conclusão de curso. O Capítulo 4 mostra, com mais detalhes, algumas das aplicações do *Max-Cut*. No Capítulo 5, são apresentadas algumas das abordagens, propostas anteriormente, para *Max-Cut*, enquanto o Capítulo 6 apresenta os algoritmos desenvolvidos neste trabalho. O Capítulo 7 apresenta os resultados obtidos pelos algoritmos juntamente com uma análise desses resultados. Por fim, no Capítulo 8 são apresentadas as conclusões e os trabalhos futuros.

2 REFERENCIAL TEÓRICO

Para a definição matemática do problema abordado nesse trabalho são necessários alguns conhecimentos iniciais na área de Teoria dos Grafos e Pesquisa Operacional que serão apresentados nas Seções 2.1 e 2.2, respectivamente.

2.1 Conceitos Necessários de Teoria dos Grafos

Um *grafo* $G = (V(G), E(G))$ consiste de um conjunto finito não vazio $V(G)$ de elementos, chamados *vértices*, e um conjunto de pares não-ordenados de elementos distintos de $V(G)$, chamado *arestas*, isto é, $E(G) \subseteq \{\{u,v\} | u,v \in V(G), u \neq v\}$. Algumas vezes, quando o grafo estiver claro pelo contexto, utilizamos V e E para $V(G)$ e $E(G)$ respectivamente, para simplificar a notação. Um corte de G, no contexto do *Max-Cut*, é qualquer bipartição $(S, \bar{S} := V \backslash S)$ dos vértices de V, ou seja, o subconjunto de arestas com uma extremidade em S e outra em $\bar{S}$, onde S ou $\bar{S}$ pode ser vazio.

A densidade de um grafo e dada em função da relação entre sua ordem (quantidade de vértices) e seu tamanho (quantidade de vértices + quantidade de arestas), ou seja, representa a proporção entre quantidade de arestas $E(G)$ e vértices $V(G)$. Essa proporção classifica os grafos em dois tipos: densos e esparsos. Seja G' um grafo completo, aquele onde cada vértice pertencente a $V(G')$ tem $(|V(G')| - 1)$ arestas conectadas a ele, quanto mais próximo $|E(G)|$ for de $|E(G')|$ mais denso esse grafo G será considerando. Quando menor o valor de $|E(G)|$ mais esparso o grafo G será considerado.

2.2 Conceitos Necessários de Pesquisa Operacional

Segundo Hillier e Lieberman (2013) a Pesquisa Operacional (PO) é um ramo interdisciplinar da matemática aplicada que faz uso de modelos matemáticos, estatísticos e de algoritmos na ajuda à tomada de decisão. É usada sobretudo para analisar sistemas complexos do mundo real, tipicamente com o objetivo de melhorar ou otimizar o desempenho. Os problemas determinísticos de PO podem ser classificados em categorias genéricas: Problemas de Programação Linear, Simulação, Teoria dos Jogos, Programação Dinâmica e Programação Não-Linear. Existem casos em que os modelos de PO são tão complexos que se torna inviável encontrar a solução ótima. Em tais situações, ainda é importante encontrar uma boa solução viável. Os métodos heurísticos são comumente usados na busca de tal solução.

2.2.1 Programação Linear

Para Luna e Goldbarg (2000) a Programação Linear apresenta algoritmos extremamente eficientes e que podem ser facilmente resolvidos utilizando o computador. A Programação Linear é uma técnica que utiliza instrumentos matemáticos que permitem a otimização de operações, e é largamente utilizada na resolução de problemas que tenham seus modelos representados por expressões lineares.

> Os problemas de Programação Linear referem-se à distribuição eficiente de recursos limitados entre atividades competitivas, com a finalidade de atender a um determinado objetivo, por exemplo, maximização de lucros ou minimização de custos. Em se tratando de programação linear, esse objetivo será expresso por uma função linear, à qual se dá o nome de função objetivo. É claro que é necessário dizer quais as atividades que consomem cada recurso, e em que proporção é feito esse consumo. Essas informações serão fornecidas por equação ou inequações lineares, uma para cada recurso. Ao conjunto dessas equações ou inequações lineares dá-se o nome de restrição do modelo. Geralmente existem inúmeras maneiras de distribuir os escassos recursos entre as diversas atividades, bastando para isso que essas distribuições sejam coerentes com as equações de consumo de cada recurso, ou seja, que elas satisfaçam as restrições do problema. Entretanto, deseja-se achar aquela distribuição que satisfaça as restrições do problema, e que alcance o objetivo desejado, isto é, que maximize o lucro ou minimize o custo. A essa solução dá-se o nome de solução ótima. Uma vez obtido o modelo linear, constituído pela função objetivo (linear) e pelas restrições lineares, a programação linear se incumbe de achar a sua solução ótima (PUCCINI, 1980).

Segundo Caixeta-Filho (2001), algebricamente, a Programação Linear é o aprimoramento de uma técnica de resolução de sistema de equações lineares, utilizando inversões sucessivas de matrizes, incorporando uma equação linear adicional representativa de um dado comportamento que deverá ser otimizado. Para (GARCIA *et al.*, 1997), matematicamente, pode-se formular o seguinte modelo de um problema de otimização de acordo com o seguinte esquema:

$$
\begin{aligned}
&\text{Maximizar ou Minimizar:} && Z = C_1X_1 + C_2X_2 + \cdots + C_nX_n && \\
&\text{Sujeito a} && a_{11}X_1 + a_{12}X_2 + \cdots + a_{1n}X_n && < b_1 \\
& && a_{21}X_1 + a_{22}X_2 + \cdots + a_{2n}X_n && < b_2 \\
& && \vdots && \\
& && a_{m1}X_1 + a_{m2}X_2 + \cdots + a_{mn}X_n && < b_m \\
& && X_i \geq 0 \quad \forall i = 1,2,\ldots n, \quad b_j \geq 0 \quad \forall j = 1,2,\ldots m &&
\end{aligned}
$$

onde:

Z = função a ser maximizada ou minimizada (geralmente ganho ou custo), respeitando o conjunto de elementos do problema ou restrições;

X_i = variáveis que representam as quantidades ou recursos que se quer determinar para otimizar o resultado global;

C_i = coeficientes de ganho ou custo que cada variável é capaz de gerar;

b_j = quantidade disponível de cada recurso;

a_{ij} = quantidade de recurso em que cada variável decisória consome.

Problemas simples ou extremamente elaborados de Programação Linear podem ser resolvidos utilizando o método simplex, algoritmo introduzido por (DANTZIG, 1953).

2.2.2 Programação Linear Inteira

Um problema de Programação Linear Inteira (PLI) se caracteriza pelo fato de conter todas as variáveis com valores discretos, ou seja, inteiras. No caso em que apenas algumas das variáveis são inteiras, o problema é chamado de Programa Linear Inteira Mista. Outro caso particular é quando as variáveis são inteiras e restritas aos valores 0 ou 1, esse caso é conhecido como Programa Linear Inteiro Binário (WOLSEY, 1998).

Mesmo sendo um grande foco de estudo em Pesquisa Operacional, ainda não foi desenvolvido um algoritmo eficiente para resolver um PLI de tamanho razoável, ou seja, não existe um algoritmo que resolva um PLI com um número de passos polinomial no tamanho da entrada. De fato, resolver um PLI é um problema da classe NP-Completo. Acredita-se que não exista solução eficiente para um problema dessa classe e qualquer solução correta do problema necessita de tempo exponencial no pior caso.

2.2.3 Desigualdades Válidas

Desigualdades Válidas são restrições que não fazem parte da modelagem inicial, mas que podem ser introduzidas no modelo como restrições adicionais. As desigualdades podem efetuar corte em regiões não factíveis, ou seja, regiões que não contém soluções inteiras admissíveis. Por exemplo, seja $X = \{x \in B^5 : 3x_1 - 4x_2 + 2x_3 - 3x_4 + x_5 \leq -2\}$, observe que:

- Se $x_2 = x_4 = 0$, então $3x_1 + 2x_3 + x_5 \leq -2$ é impossível. Consequentemente, $x_2 + x_4 \geq 1$ é uma desigualdade válida.
- Se $x_1 = 1$ e $x_2 = 0$, então $0 \leq 3 + 2x_3 + 3x_4 + x_5 \leq -2$ é impossível. Consequentemente, $x_1 - x_2 \leq 0$ é uma desigualdade válida.

A utilização de desigualdades válidas possibilita um ganho de desempenho na busca pela solução ótima, já que haverão menos soluções a serem vasculhadas. Na literatura é comum ver trabalhos

apresentando algoritmos que utilizam conjuntos de desigualdades válidas como plano de corte (SIMONE *et al.*, 1995; SIMONE; RINALDI, 1994; RENDL *et al.*, 2007).

2.2.4 Métodos Heurísticos

Para Hillier e Lieberman (2013) um método *heurístico* é um procedimento que provavelmente vai encontrar uma boa solução viável, mas não necessariamente uma solução ótima, para o problema específico ao qual ela está aplicada. Não é possível garantir a qualidade da solução obtida, porém um método heurístico geralmente é capaz de fornecer uma solução que se encontra pelo menos próxima da ótima. O procedimento normalmente é um algoritmo iterativo completo em que cada iteração envolve a condução de uma busca por uma nova solução que, eventualmente, poderia ser melhor que a melhor solução encontrada previamente. Quando o algoritmo termina após um critério de parada, seja de tempo, iterações ou um valor fixo, a solução por ele fornecida é a melhor que foi encontrada durante as iterações.

Uma *meta-heurística* é um método de resolução geral que fornece tanto uma estrutura quanto as diretrizes de estratégias gerais para desenvolver um método heurístico específico que se ajuste a um tipo de problema particular. A meta-heurística se tornou uma das mais importantes técnicas para os profissionais da área de PO (HILLIER; LIEBERMAN, 2013).

As meta-heurísticas podem ser classificadas, de maneira geral, em dois conjuntos: meta-heurísticas singulares, tais como SA, *Greedy Randomized Adaptive Search Procedure* (GRASP), *Variable Neighborhood Search* (VNS) e Busca em Tabu, onde esses algoritmos abordam uma única solução e sua estratégia de busca consiste em explorar a vizinhança da solução utilizada. Segundo conjunto, meta-heurísticas populacionais, tais como Algoritmos Genéticos, Colônia de Formigas e Colônia de Abelhas, utilizam populações, ou seja, um conjunto de soluções, diferente das meta-heurísticas singulares, e seu poder de busca está em modificar essa população a cada iteração e selecionar os indivíduos que se manterão, descartando o restante.

2.2.4.1 Algoritmo Genético

Segundo Mitchell (1998) o Algoritmo Genético (AG) é uma meta-heurística que se baseia em uma analogia de um fenômeno natural. Nesse caso, a analogia é a teoria da evolução biológica formulada por Charles Darwin. É uma simulação computacional iterativa que faz analogia ao processo evolutivo de várias gerações de uma população, onde cada indivíduo é uma representação abstrata de um solução do problema, a seleção natural é um critério de escolha das

melhores soluções e eliminação das ruins, o cruzamento e a mutação são meios para obtenção de novas soluções.

Para cada iteração (geração) de um AG, a *população* atual é formada pelo conjunto de soluções experimentais que estão sendo consideradas. Essas soluções experimentais são imaginadas como os atuais membros viventes da espécie. Alguns dos membros mais adaptados sobrevivem, passam para a idade adulta e se tomam *pais*, formando pares aleatórios, que depois têm *filhos*. Esses filhos são novas soluções experimentais que compartilham algumas das características de ambos os pais.

Já que os membros mais adaptados da população têm mais chance de se tornar pais que outros, um AG tende a gerar populações melhores de soluções experimentais à medida que prossegue. Ocasionalmente, ocorrem *mutações* de modo que certos filhos também adquirem características que nenhum dos pais possui. Isso ajuda um AG a explorar uma nova região de soluções viáveis. Finalmente, a sobrevivência dos mais adaptados levará um AG a uma solução experimental que é próxima da solução ótima. Um pseudocódigo do AGC é apresentado a seguir:

Algoritmo 1: Algoritmo Genético Clássico

Saída: Melhor solução encontrada

início

    Inicialize a população;

    Enquanto *critério de parada não satisfeito* **Faça**

        Selecione indivíduos da população para reprodução;

        Aplique os operadores Genéticos;

        Avalie os indivíduos da população;

        Selecione indivíduos para sobreviver;

    Fim

fim

Alguns dos parâmetros mais comuns para o AGC podem ter grande influência nos resultados obtidos, por exemplo o tamanho da população inicial e a quantidade de novos indivíduos que serão adicionado a cada iteração, ou seja, geração do algoritmo. É possível ressaltar que os critérios de parada mais comuns são os de tempo de execução máxima, quantidade máxima de gerações ou quantidade máxima de iterações sem que o algoritmo obtenha melhoras na população. Já para o procedimento de seleção dos indivíduos para reprodução, é possível selecionar indivíduos de maneira aleatória, com maior aptidão, mesclando entre indivíduos de

alta e baixa aptidão e a abordagem conhecida como torneio, onde são sorteados alguns indivíduos e o melhor deles, ou seja, o com o maior valor de aptidão, é selecionado.

A parte de operadores genéticos consiste principalmente nas funções de cruzamento e mutação, que podem ser aplicadas a todos os novos indivíduos gerados ou cada uma das funções pode ter uma taxa de aplicação. Algumas versões do AG contém diferentes funções de cruzamento e mutação para os indivíduos, nestes casos é comum a utilização de uma técnica de roleta, que consiste em sortear, de maneira aleatória ou não, qual função será aplicada. Para a seleção final da população, vale ressaltar que é possível manter indivíduos com aptidão muito baixa, não necessariamente mantém-se apenas os indivíduos mais aptos, essa mistura de indivíduos contribui para a variabilidade de soluções do AG.

2.2.4.2 Têmpera Simulada

A abordagem de Têmpera Simulada, também conhecida na literatura estrangeira como *Simulated Annealing* SA é uma meta-heurística que permite ao processo de busca escapar de um ótimo local. Essa meta-heurística é uma metáfora de um processo térmico, dito *annealing* ou recozimento, utilizado em metalurgia para obtenção de estados de baixa energia num sólido. Um processo físico de têmpera acontece quando um sólido em um banho térmico é esquentado inicialmente mediante o incremento da temperatura, tal que os esforços térmicos são liberados e todas as partículas são distribuídas aleatoriamente numa fase liquida. Isso é seguido por um esfriamento lento até a temperatura do ambiente, de forma que todas as partículas arranjam-se em um estado de baixa energia no qual a solidificação acontece.

Sendo mais específico, segundo Hillier e Lieberman (2013) cada iteração do SA busca deslocamentos da solução experimental atual para um vizinho imediato na vizinhança local dessa solução. Esta técnica começa a busca a partir de uma solução inicial qualquer e para cada solução vizinha gerada é feita um cálculo, baseado na temperatura atual, que indica se a solução vizinha substituirá a atual solução ou não, ao final é efetuada uma redução da temperatura. Um pseudocódigo do SA é apresentado a seguir no Algoritmo 2.

Um dos parâmetros mais importantes do SA é o de temperatura máxima, uma vez que ele influencia diretamente na probabilidade de escolha da solução, ou seja, quanto maior a temperatura maior a probabilidade da solução vizinha ser aceita, por mais que ela contenha menor energia que a solução atual. A temperatura máxima ainda pode ser utilizada como um dos critérios de parada, uma vez que a cada iteração do SA ela é reduzida, é possível utilizar também

como critério de parada um tempo máximo de execução.

A característica mais importante do SA está na probabilidade de aceitação da solução vizinha, pois com isso é possível que o algoritmo possa aceitar soluções consideradas ruins, ou seja, com menor valor de energia. Essas soluções ruins fazem com que o algoritmo tenha a oportunidade de sair do que são considerados ótimos locais e possa explorar uma nova região de soluções do problema.

Algoritmo 2: Têmpera Simulada

Saída: Melhor solução encontrada

início

$x^a \leftarrow$ solução aleatória inicial;

$T \leftarrow$ temperatura máxima;

Enquanto *Critério de parada não satisfeito* **Faça**

$x^n \leftarrow$ solução vizinha de x^a;

$\Delta \leftarrow$ Energia(x^n) $-$ Energia(x^a);

$u \leftarrow$ valor aleatório entre 0 e 1;

Se *(*$\Delta < 0$*) ou (*$e^{-\Delta/T} > u$*)* **Então**

$x^a \leftarrow x^n$;

Fim

Reduzir o valor de T;

Fim

fim

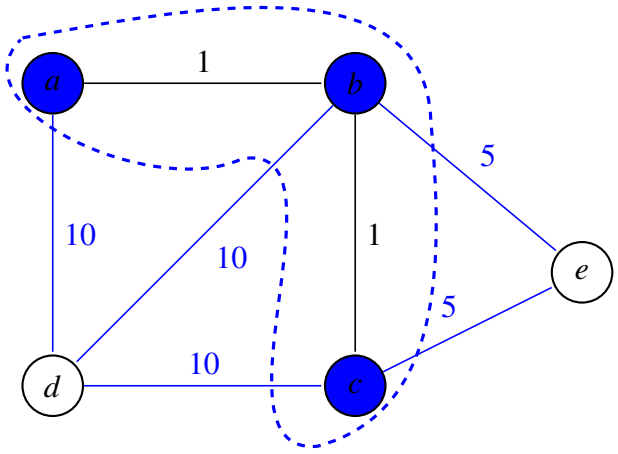
3 PROBLEMA DO CORTE MÁXIMO

Este Capítulo descreve com mais detalhes algumas propriedades relevantes do problema proposto nesta monografia. A seção 3.1 apresenta uma descrição do *Max-Cut*. Algumas das formulações para o *Max-Cut* são apresentadas nas Seções 3.2.1, 3.2.2 e 3.2.3. A Seção **??** apresenta os diferentes tipos de instâncias que são aplicadas ao *Max-Cut* e as Desigualdades Válidas que serão aplicadas ao *Max-Cut* no presente trabalho serão provadas na Seção 3.3.

3.1 Max-Cut

Max-Cut pode ser descrito da seguinte forma: dado um grafo não direcionado $G = (V, E)$ com $|V| = n$ vértices e $|E| = m$ arestas ponderadas. Qualquer bipartição, representada por $(S, \bar{S})$, dos vértices de V define um corte de G, ou seja, o subconjunto de arestas com uma extremidade em S e a outra em $\bar{S}$, onde S ou $\bar{S}$ pode ser vazio. O *Max-Cut* consiste em encontrar $(S, \bar{S})$ tal que a soma dos pesos das arestas do corte seja o maior possível. O *Max-Cut* possui muitas formulações equivalentes, as quais aparecem com diferentes nomes na literatura, além de *Max-Cut* as denominações mais comuns são: *Maximum-2-Satisfiability, Weighted Signed Graph Balancing* e *Unconstrained Quadratic 0-1 programming* (HAMER, 1991). Um exemplo de Corte Máximo é apresentado na Figura 1.

Figura 1 – (a) O corte que maximiza o valor de soma das arestas que estão entre os subconjuntos S (Nós azuis) e $\bar{S}$ (Nós brancos).



Fonte: Elaborado pelo autor (2018).

3.2 Formulações

3.2.1 Formulação Binária {0, 1}

Se for apresentada uma bipartição $(S, \bar{S})$ pelo vetor binário $x \in \{0,1\}$ com $x_i = 1$ se $i \in S$ e $x_i = 0$ caso contrário, então o problema do *Max-Cut* pode ser formulado como um problema sem restrições, com uma função objetivo quadrática e suas variáveis assumindo valores binários, onde $x \in \{0,1\}$ (BARAHONA, 1983). Por simplicidade de escrita:

$$\begin{aligned} MC_I = \max & \sum_{i=1}^{n} \sum_{j=i+1}^{n} c_{ij}(x_i + x_j - 2x_i x_j) \\ s.a & \ \ x \in \{0,1\}^n \end{aligned} \tag{3.1}$$

A formulação (3.1) pode ser escrita de outra forma. A formulação a seguir será tomada como base para obtenção das desigualdades válidas apresentadas na Seção 3.3.

$$\begin{aligned} MC_{II} = \max & \sum_{i=1}^{n} \sum_{j=i+1}^{n} c_{ij}(x_i \bar{x_j} + \bar{x_i} x_j) \\ s.a & \ \ x \in \{0,1\}^n,\ \bar{x} = 1 - x_j \ \forall j = 1, \ldots, n \end{aligned} \tag{3.2}$$

3.2.2 Formulação Binária {-1, 1}

Uma outra forma de enxergar e formular o *Max-Cut* se dá quando é considerado o seguinte vetor binário $x \in \{1,-1\}^n$, com $x_i = 1$ se $i \in S$ e $x_i = -1$ se $i \in \bar{S}$ (GOEMANS; WILLIAMSON, 1995b). Assim podemos modelar o *Max-Cut* da seguinte maneira:

$$\begin{aligned} MC_{III} = \max & \sum_{i=1}^{n} \sum_{j=i+1}^{n} c_{ij} \left(\frac{1 - x_i x_j}{2} \right) \\ s.a & \ \ x \in \{1,-1\}^n \end{aligned} \tag{3.3}$$

3.2.3 Demais Formulações

Existem variações do modelo quadrático que derivam das Formulações (3.2.1) e (3.2.2). Também estão presentes na literatura diversas formulações lineares e de programação semidefinida para o *Max-Cut*, ver (SOARES *et al.*, 2017; GOEMANS; WILLIAMSON, 1995b; LAURENT *et al.*, 1997). Essas linearizações não serão abordadas neste trabalho, pois os modelos (3.2.1) e (3.2.2) serão suficientes para o entendimento das Desigualdades Válidas e das aplicações descritas na Seção 4.

3.3 Condições de Otimalidade

Nesta seção será considerada a formulação para o *Max-Cut* expressa em (3.2) em termos da variável $x \in \{0,1\}$, onde x_i marca a partição à que o vértice i pertence. A função objetivo é $Q(x) = \sum_{i \in V} \sum_{\substack{j \in V \\ j>i}} c_{ij} \left[x_i(1-x_j) + (1-x_i)x_j \right]$. Note que

$$Q(x) = \sum_{i \in V} \sum_{\substack{j \in V \\ j \neq i}} c_{ij} x_i - \sum_{i \in V} \sum_{\substack{j \in V \\ j \neq i}} c_{ij} x_i x_j \tag{3.4}$$

$$= \sum_{i \in V} \sum_{\substack{j \in V \\ j \neq i}} c_{ij} x_i (1-x_j) \tag{3.5}$$

Dados $x \in \{0,1\}^n$ e $k \in \{1,2,\ldots,n\}$, seja $x^{\bar{k}} \in \{0,1\}^n$ tal que

$$x_i^{\bar{k}} = \begin{cases} x_i, & i \neq k \\ 1-x_i, & i = k \end{cases} \tag{3.6}$$

Defina $C_k = \sum_{\substack{j \in V \\ j \neq k}} c_{kj}$ e $C_k(x) = \sum_{\substack{j \in V \\ j \neq k}} c_{kj} x_j$. Determinamos a seguir a relação entre os valores de $Q(x)$ e $Q(x^{\bar{k}})$.

A maneira com que as restrições foram codificadas neste trabalho pode ser representada no algoritmo 3. De modo simplificado, é possível descrever os componentes do conjunto de desigualdades da seguinte maneira:

- C é um vetor onde cada posição representa um vértice do grafo, com isso, o valor da posição k, do vetor, é igual ao somatório dos pesos de todas as arestas que conectam os vértices k e j, tal que, $j \in V(G)$ e $k \neq j$.
- $C_k(x)$ é um número real calculado a partir do vetor binário de solução x. Assim, o valor de $C_k x$ é o somatório de todas as arestas que conectam os vértices k e j, tal que, $j \in V(G)$ e $k \neq j$ e cada aresta, antes de ser somada, é multiplicada pelo valor de x_j, ou seja, por 1 se o vértice j está no grupo S e 0 caso contrário. Nota-se o fato de que o grupo onde cada vértice está influencia diretamente no valor obtido ao final do somatório, por esse motivo o calculo de $C_k(x)$, para ser utilizado, precisa ser refeito a cada alteração na solução corrente.

Lema 3.3.1 *Para todo $x \in \{0,1\}^n$ e todo $k \in \{1,2,\ldots,n\}$, $Q\left(x^{\bar{k}}\right) - Q(x) = \sum_{\substack{j \in V \\ j \neq k}} c_{kj}(1-2x_k)(1-2x_j)$*

Demonstração.

$$\begin{aligned} Q\left(x^{\bar{k}}\right) - Q(x) &= \sum_{\substack{j\in V\\ j\neq k}} c_{kj}\left[x_k^{\bar{k}}(1-x_j^{\bar{k}}) - x_k(1-x_j)\right] + \sum_{\substack{i\in V\\ i\neq k}} c_{ik}\left[x_i^{\bar{k}}(1-x_k^{\bar{k}}) - x_i(1-x_k)\right] \\ &= \sum_{\substack{j\in V\\ j\neq k}} c_{kj}\left[(1-x_k)(1-x_j) - x_k(1-x_j)\right] + \sum_{\substack{i\in V\\ i\neq k}} c_{ik}\left[x_i x_k - x_i(1-x_k)\right] \\ &= \sum_{\substack{j\in V\\ j\neq k}} c_{kj}(1-2x_k)(1-x_j) + \sum_{\substack{i\in V\\ i\neq k}} c_{ki}x_i(-1+2x_k) \\ &= \sum_{\substack{j\in V\\ j\neq k}} c_{kj}(1-2x_k)(1-2x_j) \end{aligned}$$

■

O lema acima leva as seguintes considerações para a fixação de uma variável em uma solução ótima.

Corolário 1 *Sejam x^* uma solução ótima para o Max-Cut e $k \in \{1,2,\ldots,n\}$. Se $x_k^* = 1$ então $C_k(x^*) \leq C_k/2$. Por outro lado, se $C_k(x^*) < C_k/2$ então $x_k^* = 1$.*

Demonstração.

Defina $\bar{x} = (x^*)^{\bar{k}}$. Pelo Lema 3.3.1

$Q(\bar{x}) - Q(x^*) = \sum_{\substack{j\in V\\ j\neq k}} -c_{kj}(1-2x_j^*) = -\sum_{\substack{j\in V\\ j\neq k}} c_{kj} + \sum_{\substack{j\in V\\ j\neq k}} c_{kj}x_j^* = -C_k + 2C_k(x^*) > 0.$

Logo $Q(\bar{x}) > Q(x^*)$: um absurdo pois x^* é ótimo.

Por outro lado, considere que $C_k(x^*) < C_k/2$ e suponha, por contradição que $x_k^* = 0$.

Novamente pelo Lema 3.3.1 temos

$Q(\bar{x}) - Q(x^*) = \sum_{\substack{j\in V\\ j\neq k}} c_{kj}(1-2x_j^*) = C_k - 2C_k(x^*) > 0$*, levando também à um absurdo.*

■

Corolário 2 *Sejam x^* uma solução ótima para o Max-Cut e $k \in \{1,2,\ldots,n\}$. Se $x_k^* = 0$ então $C_k(x^*) \geq C_k/2$. Por outro lado, se $C_k(x^*) > C_k/2$ então $x_k^* = 0$.*

A demonstração do Corolário 2 segue a mesma ideia da demonstração feita para o Corolário 1.

4 APLICAÇÕES

Neste capítulo são apresentadas algumas das aplicações mais comuns do *Max-Cut*. Na Seção 4.1 é apresentada uma descrição do modelo de *Spin Glass*, a Seção 4.2 contextualiza design de circuitos VLSI e em 4.3 é apresentada a utilização do *Max-Cut* em segmentação de imagens. Outras aplicações tais como análise de rede social, geração de teste de falha e atribuição de multiprocessadores em redes distribuídas são apresentadas em (BOROS; HAMMER, 1991); (POLJAK; TUZA, 1993).

4.1 Spin Glasses

O modelo Ising, de *Spin Glass*, usa elétrons de spin para determinar o potencial energético de um *array* de átomos. Quando esses átomos são organizados em estruturas cristalinas, uma direção de giro dominante pode aparecer, com isso, o vetor $s \in \{1,-1\}^n$ modela os elétrons de spin desse cristal, onde $s_i = 1$ se ele é paralelo a direção dominante e $s_i = -1$ se ele é antiparalelo. Seja W uma matriz de interações de giro, representando vizinhos próximos, o valor de h é determinado pela presença de um campo magnético externo, portando, a matriz Hessiana (quadrada), H do cristal é dada por:

$$H(s) = -\sum_{i=1}^{n}\sum_{j=1}^{n} J_{ij}s_i s_j - h\sum_{i=1}^{n} s_i \tag{4.1}$$

Este modelo é um exemplo onde cada i é um vértice e os valores de s_i são iguais, se $i \in S$ ou $i \in V \setminus S$. A introdução de um vértice fictício s_0 nos permite reduzir o segundo somatório na equação (4.1) para o primeiro com $J_{00} = 0$ e $J_{i0} = J_{0i} = \frac{h}{2}$ quando $i \neq 0$, resultando assim no seguinte modelo:

$$H(s) = -\sum_{i=0}^{n}\sum_{j=0}^{n} J_{ij}s_i s_j. \tag{4.2}$$

Assim, o problema para encontrar a energia do estado fundamental de um modelo de *Spin Glass* é o mesmo que encontrar o máximo,

$$\max_{s\in\{1,-1\}^n} H(s) = \sum_{i=0}^{n}\sum_{j=0}^{n} J_{ij}s_i s_j = \max_{s\in\{1,-1\}^n} s^t J_s \tag{4.3}$$

A comparação entre o lado esquerdo da equação (4.3) e a formulação apresentada em (3.2.2) nos mostra que a solução para o *Max-Cut* é a mesma para o Problema de *Spin Glass* (HARTMANN; RIEGER, 2006).

4.2 VLSI

Integração em escala muito grande (*Very Large Scale Integrated* - VLSI) é o *design* e o leiaute dos circuitos integrados (LIERS *et al.*, 2011). O encaminhamento dos fios ocorre após a colocação dos elementos do circuito e das redes que permitem que os fios conflitantes passem uns pelos outros em diferentes camadas. Um dos objetivos do encaminhamento é minimizar o número de locais onde os fios mudam de camada, já que isso é feito com vias, pequenos furos feitos no chip, um processo que arrisca a quebra do chip e o aumento dos custos. As redes permitem que os fios passem entre elas estabelecendo segmentos críticos para cada um dos fios. Segmentos críticos não permitem vias. As vias entre as redes são permitidas nos chamados segmentos livres.

Para gerar o grafo que será resolvido pelo *Max-Cut* deve ser feito um mapeamento no chip. O grafo $G = (V, E)$ será baseado nos segmentos críticos e livres (BARAHONA *et al.*, 1988). Cada vértice representa o tamanho do fio em uma seção crítica e as arestas são divididas em dois tipos. O primeiro tipo de aresta, chamada de aresta de conflito, conecta vértices que representam fios que estão se cruzando no segmento, isto é, segmentos em conflito. O segundo tipo de aresta, a aresta livre, conecta segmentos de fios críticos de acordo com os segmentos livres, ou seja, a via pela qual o fio passa para outra camada.

As arestas de G podem ser valoradas da seguinte maneira, arestas de conflito tem peso positivo, já as arestas livres tem peso 0 (ou negativo). Desse modo, podemos minimizar a quantidade de vias aplicando um corte em G. Este corte conterá o máximo de arestas de conflito e, por conseguinte, minimiza o número de arestas livres no corte. Quanto mais arestas de conflito forem selecionadas, maior será a quantidade de segmentos críticos nas redes, e com isso, menor será o espaço que permite a criação de vias.

4.3 Segmentação de Imagens

Segmentação de imagens se refere ao processo de dividir uma imagem digital em múltiplas regiões (conjunto de pixeis), com o objetivo de simplificar e/ou mudar a representação

de uma imagem para facilitar a sua análise. Segmentação de imagem é tipicamente usada para localizar objetos e formas (linhas, curvas, etc) em imagens. O resultado da segmentação de imagens é um conjunto de regiões ou um conjunto de contornos extraídos da imagem. Desse modo, cada um dos pixeis em uma mesma região é similar com referência a alguma característica ou propriedade computacional, tais como cor, intensidade, textura ou continuidade. Regiões adjacentes devem possuir diferenças significativas com respeito a mesma característica(s).

Nessa aplicação, a imagem é dividida em regiões de pixeis adjacentes com cores similares, e cada região é representada por um vértice no Grafo G. O tamanho da região está diretamente relacionado com o tipo de vizinhança selecionada para os pixeis (4-Conectado, 8-conectado etc). Em Sousa *et al.* (2013) cada vértice tem um peso, que é o valor da média de similaridade das cores presentes nessa região. Assim, para poder aplicar o *Max-Cut*, serão criadas arestas e_{ij}, que conectam os vértices v_i e v_j, tal que $v \in V(G)$. O peso de e_{ij} é calculado pela diferença dos valores de x_i e x_j, quanto maior a diferença, maior será o valor atribuído a e. Após solucionar o *Max-Cut* no grafo que corresponde a imagem, as regiões que não foram separadas pelo corte serão misturadas em uma única região maior, produzindo uma imagem segmentada.

5 REVISÃO DA LITERATURA

Utilizando relaxação em programação semidefinida, Goemans e Williamson (1995a) foram os primeiros a dar esperança de resolução para instâncias densas, ao mostrarem um algoritmo aproximativo que sempre oferece uma solução de pelo menos 0.87856 vezes o valor da solução ótima para o *Max-Cut*, desde que os pesos das arestas sejam não negativos.

Diversas abordagens heurísticas e meta-heurísticas têm sido propostas na literatura para obter solução para o *Max-Cut*. Kim *et al.* (2001) abordaram o *Max-Cut* através de algoritmos híbridos baseados em AG. Essa abordagem usa instâncias reais, derivadas de Circuitos VLSI 4.2. Em Mansour *et al.* (2006) o *Max-Cut* é resolvido utilizando um Algoritmo Genético Incremental (AGI). Ambos os trabalhos não comparam os resultados obtidos com o estado da arte para *Max-Cut*, as comparações se limitam as diferentes versões propostas em cada um dos trabalhos.

Heurísticas baseadas em GRASP. VNS e *Variable Neighborhood Descent* (VND) são apresentadas em (FESTA *et al.*, 2001; FESTA *et al.*, 2002). Essas abordagens conseguem resultados em pequena quantidade de tempo, no entanto, esses resultados são inferiores em relação aos obtido pelos modelos formulados com programação semidefinida.

Em Wu e Hao (2012) é proposto um algoritmo memético, pertencente ao conjunto dos algoritmos evolutivos, que consegue ser competitivo com alguns dos melhores algoritmos propostos para o *Max-Cut*. Em Kochenberger *et al.* (2013) é relatado um novo algoritmo de Busca em Tabu que consegue bons resultados para instâncias com grande quantidade de vértices. Ambos os métodos forneceram algumas das melhores soluções conhecidas para instâncias de teste utilizadas na literatura.

Em Dunning *et al.* (2018) é feita uma revisão sistemática acerca dos algoritmos heurísticos, propostos na literatura, para o *Max-Cut* e para o problema de Otimização Quadrática Binária sem Restrições (QUBO). Foram replicados um total de 37 algoritmos heurísticos, que estão disponibilizados em [1]. Esses algoritmos foram alimentados com mais de 2000 instâncias, de tamanhos e densidades variadas. A performance dos algoritmos serviram de dados de entrada para modelos de aprendizado de máquina. É possível observar que os algoritmos evolutivos obtiveram um melhor desempenho para instâncias densas de tamanho mediano, enquanto os algoritmos não evolutivos apresentaram um melhor desempenho para instâncias de tamanho grande.

Diante da análise feita a partir de trabalhos anteriores, é notório a extensa utilização

[1] <https://github.com/MQLib>

de abordagens heurísticas para obtenção de soluções para o *Max-Cut*, fato que motiva a criação de abordagens híbridas que otimizem esses algoritmos, uma vez que, além de inexistentes na literatura, meta-heurísticas que utilizem desigualdades válidas podem ser uma nova maneira de efetuar modificações menos aleatórias na solução. Em contrapartida, é difícil encontrar e provar desigualdades válidas, e ainda, levando em consideração as que já existem, destaca-se a dificuldade para aplicação eficiente em problemas de otimização.

6 ALGORITMOS PROPOSTOS

6.1 Codificação das Desigualdades Válidas

Usando como exemplo o grafo mostrado na Figura 1, é possível representar essa solução como um vetor binário, onde cada posição do vetor é equivalente a um vértice da solução e o valor contido na posição do vetor representa o grupo ao qual o vértice pertence, 0 ou 1. A Figura 2 exibe o indivíduo, neste caso por ser uma solução ótima, pode-se trocar o nome *indivíduo* por x^*. Logo abaixo está o vetor C_k e os respectivos valores de $C_k(x^*)$ para cada vértice. No último vetor as propriedades para os subconjuntos, 0 e 1, são apresentados.

Figura 2 – Representação do comportamento das desigualdades válidas para a solução ótima da Figura 1.

	1	2	3	4	5
indivíduo =	0	0	0	1	1
C_k =	11	17	16	30	10
$C_k(x^*)$ =	10	15	15	0	0

$C_k(x^*) > C_k/2$	$C_k(x^*) < C_k/2$

Fonte: Elaborado pelo autor (2018).

A criação de indivíduos utilizando as desigualdades válidas é representado no Algoritmo 3. Consiste basicamente em inicializar o indivíduo com o valor 0 em todas os seus genes, ou seja, o vetor de solução que ele representa e depois ir modificando os valores dos genes até que toda a solução esteja de acordo com o conjunto de restrições. Para os casos em que as desigualdades forem utilizadas na modificação de uma solução, haverá a diferença na entrada, pois adicionaremos na entrada o indivíduo que será alterado e a fase de modificar para o valor 0 em todos os genes é retirado.

Vale ressaltar que a fixação do valor de uma posição do vetor de solução, seja com o valor 0 ou 1, contribui para a utilização das desigualdades uma vez que é possível notar que a solução ótima para o *Max-Cut* consiste na bipartição do grafo nos conjuntos $(S, \bar{S})$, com isso, pode-se afirmar que se for gerada uma nova solução com a inversão dos subconjuntos, ou seja, $S^* \leftarrow \bar{S}$ e $\bar{S}^* \leftarrow S$, a nova solução $(S^*, \bar{S}^*)$ contém o mesmo valor ótimo de corte, mas é uma

solução diferente. Assim, é garantido que um, e somente um, vértice possa ser fixado, ao início da solução, em um subconjunto sem que isso inviabilize a obtenção da solução ótima para o *Max-Cut*.

Algoritmo 3: Criação de indivíduos utilizando as Desigualdades

Saída: Individuo
Entrada: Grafo: G
// W_{ij} Representa o valor da aresta que conecta os vértices i e j
para $k \in V(G)$ **faça**
  $individuo_k \leftarrow 0$;
Fim
para $k \in V(G)$ **faça**
  para $j \in V(G)$ **faça**
    Se $k \neq j$ **Então**
      $C_k \leftarrow C_k + W_{kj}$;
    Fim
  Fim
Fim
fixado $\leftarrow$ posição aleatória do indivíduo;
Enquanto *Existe* $individuo_k$ *tal que* $individuo_k = 0$ & $C_k(x) < C_k/2$ *ou* $individuo_k = 1$ & $C_k(x) > C_k/2$ **Faça**
  para $k \in individuo$ **faça**
    Se $individuo_k \neq fixado$ **Então**
      para $j \in individuo$ **faça**
        Se $k \neq j$ **Então**
          $C_k(x) \leftarrow C_k(x) + (W_{kj} * individuo_j)$;
        Fim
      Fim
      Se $C_k(x) \leq C_k/2$ **Então**
        $individuo_k \leftarrow 1$;
      Senão
        $individuo_k \leftarrow 0$;
      Fim
    Fim
  Fim
Fim

6.2 Algoritmo Genético - Desigualdades Válidas na Geração da População Inicial

Nesta seção será apresentada a descrição da versão do Algoritmo Genético que utiliza as desigualdades na primeira etapa do algoritmo, ou seja, na geração da população inicial. Essa heurística, que será denominado como AG-PI (PI - População Inicial), está representada

no Algoritmo 4. O *AG-PI* cria a população da seguinte maneira: primeiramente são fixados alguns vértices no grupo *S*, ou seja, grupo 1, e esses vértices, apenas na etapa de criação, não serão movidos para outro grupo pelas restrições, o intuito da fixação dos vértices é fazer com que haja uma diversidade nos indivíduos gerados. O restante do algoritmo segue o padrão do AGC, apresentado na Seção 2.2.4.1.

Algoritmo 4: GA-PI

```
Saída: Melhor solução encontrada
início
    população ← ∅;
    para i ← 1 até i ≤ tamanho da população faça
        população ← população ∪ indivíduo com solução gerada pelas restrições;
    Fim
    Enquanto Critério de parada não satisfeito Faça
        individuos ← individuosParaReprodução(populaçao);
        população ← população ∪ Cruzamento(individuos);
        para individuo ∈ população faça
            individuo ← Mutação(individuo);
        Fim
        população ← Sobreviventes(população);
    Fim
fim
```

6.3 Algoritmo Genético - Desigualdades Válidas Como Função de Mutação

Esta seção descreve a versão do AG que utiliza as desigualdades como função de *Mutação*, ou seja, as soluções geradas pela função de *Cruzamento* são modificadas de maneira que sejam totalmente aceitas pelas desigualdades. Essa versão da heurística, denominada AG-M (M - Mutação), está representada no Algoritmo 5. O *AG-M* cria a população inicial com soluções aleatórias e aplica os operadores genéticos sobre essa população realizando a substituição da função de *Mutação* do AGC, por uma mutação realizada com as desigualdades válidas.

Algoritmo 5: GA-M

Saída: Melhor solução encontrada

início

população $\leftarrow \emptyset$;

para *i* $\leftarrow$ *1 até i* $\leq$ *tamanho da população* **faça**

população $\leftarrow$ população $\cup$ indivíduo com solução aleatória;

Fim

Enquanto *Critério de parada não satisfeito* **Faça**

individuos $\leftarrow$ individuosParaReprodução(populaçao);

para *individuo* $\in$ *individuos* **faça**

individuo $\leftarrow$ MutaçãoComDesigualdades(individuo);

Fim

população $\leftarrow$ população $\cup$ Cruzamento(individuos);

população $\leftarrow$ Sobreviventes(população);

Fim

fim

6.4 Algoritmo Genético - Desigualdades Válidas em Ambas as Etapas

Esta seção descreve a terceira e última versão, baseada em AG, que utiliza as desigualdades válidas numa abordagem híbrida para geração da população inicial e como uma função de mutação adicional, essa versão intitulada de *AG-HF* (HF - Híbrido Final), está representada no Algoritmo 6. A primeira etapa do *AG-HF* consiste na geração da população inicial, onde apenas 1 indivíduo da população é criado com as desigualdades, os demais são criados aleatoriamente. Essa criação da população tem como objetivo tirar proveito de uma boa solução inicial, que será fornecida pelo indivíduo criado com as desigualdades, em conjunto com uma alta variabilidade das soluções, fornecidas pelos indivíduos gerados aleatoriamente, sendo assim, o aumento da quantidade de indivíduos criados com as desigualdades influenciará na variabilidade genética da população inicial. A segunda etapa consiste em aplicar as desigualdades em todos os novos indivíduos, gerados pela função de Cruzamento. Após a seleção dos indivíduos sobreviventes, é aplicada a função de mutação do AGC na população sobrevivente.

Algoritmo 6: GA - HF

Saída: Melhor solução encontrada

início

população ← população ∪ individuo com desigualdades;

para *i* ← *2 até i* ≤ *tamanho da população* **faça**

população ← população ∪ indivíduo com solução aleatória;

Fim

Enquanto *Critério de parada não satisfeito* **Faça**

individuos ← individuosParaReprodução(populaçao);

individuos' ← Cruzamento(individuos);

para *individuo' ∈ individuos'* **faça**

individuo' ← MutaçãoComDesigualdades(individuo');

Fim

população ← população ∪ individuos';

população ← Sobreviventes(população);

para *individuo ∈ população* **faça**

individuo ← Mutação(individuo);

Fim

Fim

fim

6.5 Têmpera Simulada - Desigualdades Válidas como Modificador da Solução Corrente

Esta seção descreve a versão do SA que utiliza as desigualdades válidas como modificador da solução, essa versão, denominada como *SA-H* (H - Híbrido), é baseada no implementação do SA apresentada em (MYKLEBUST, 2015), que contém, atualmente, resultados que são competitivos com o estado da arte, do *Max-Cut*, para algumas instâncias da literatura. Com isso, a diferença entre o SA desenvolvido neste trabalho se dá pela aplicação do conjunto de desigualdades para modificar a solução corrente sempre que o algoritmo estiver, à uma certa quantidade de iterações, sem aceitar modificações para soluções vizinhas. Um código utilizando a linguagem de programação *C++* é apresentado no trabalho de (MYKLEBUST, 2015), mas para ilustrar de forma simplificada as alterações efetuadas no código para a adição das restrições, foi desenvolvido o Algoritmo 7. Os valores dos parâmetros utilizados para o *SA-H* são os mesmos apresentados no trabalho de (MYKLEBUST, 2015).

Algoritmo 7: SA-H

Saída: Melhor solução encontrada

início

$x \leftarrow$ solução aleatória inicial;

$T_{max} \leftarrow 10000$;

$calor \leftarrow 0$;

$fixo \leftarrow$ posição aleatória do vetor de solução;

$rejeita \leftarrow 0$;

Enquanto $calor < T_{max}$ **Faça**

$k \leftarrow$ posição aleatória e diferente de $fixo$;

Se *Probabilidade de aceitar solução* **Então**

$x \leftarrow$ AlteraVérticeDaSolução(x, k);

Senão

$rejeita \leftarrow rejeita + 1$;

Se $rejeita \geq$ *valor pré-definido* **Então**

$x \leftarrow$ ModificaçãoComDesigualdades(x);

$rejeita \leftarrow 0$;

Fim

Fim

$calor \leftarrow calor + 2e^{-6}$;

Fim

fim

7 RESULTADOS COMPUTACIONAIS

7.1 Configuração do Ambiente Computacional

Nesta seção são descritos os experimentos computacionais realizados para testar a eficiência das heurísticas desenvolvidas neste trabalho. A implementação dos algoritmos foi feita utilizando a linguagem de programação *C++*. Os experimentos realizados foram executados utilizando uma máquina com processador Intel Core I5-4570 (3.20GHz) 4 Threads com 14.6 GB RAM e sistema operacional Linux Mint 18 Cinnamon 64 bits.

7.2 Instancias

7.2.1 Características das Instâncias

Os experimentos foram realizados sobre uma amostra de 102 instâncias, esses conjuntos de instâncias de *benchmark* são disponibilizados em <http://www.optsicom.es/maxcut/> e <http://biqmac.uni-klu.ac.at/biqmaclib.html#ref>. A seguir apresentamos os principais dados das 102 instâncias que são divididas em 2 subconjuntos:

- *G*: Esse conjunto de instâncias geradas por (HELMBERG; RENDL, 2000) que usaram *rudy*, uma máquina independente geradora de grafos por Giovanni Rinaldi, para criar 54 instâncias variando de $n = 800$ à 3000. Ele contém grafos toirodais, planares e randômicos com as arestas assumindo os valores 1, 0 ou -1.
- *statistical physics application*: Esse conjunto de instâncias geradas por (LIERS, 2004), (LIERS *et al.*, 2004) é derivado de aplicações em física estatística e contém 48 instâncias que são divididas em 4 subconjuntos:
  - *t2gn_seed*: Para cada dimensão, três grafos de grades toroidais bidimensionais com arestas contendo valores que seguem uma distribuição Gaussiana e dimensões $n \times n$, $n = 10, 15, 20$.
  - *t3gn_seed*: Para cada dimensão, três grafos de grades toroidais tridimensionais com arestas contendo valores que seguem uma distribuição Gaussiana e dimensões $n \times n$, $n = 5, 6, 7$.
  - *ising2.5n_seed*: Para cada dimensão, três instâncias de cadeia de Ising unidimensionais. $n = 100, 150, 200, 250, 300$.
  - *ising3.0n_seed*: Para cada dimensão, três instâncias de cadeia de Ising unidimensio-

nais. $n = 100, 150, 200, 250, 300$.

O objetivo das comparação é apresentar qual dos algoritmo obteve o melhor desempenho, considerando o valor da função objetivo e o tempo de execução. Por questão de organização, os resultados serão apresentados em subseções agrupando todos os dados necessários de acordo com a abordagem, ou seja, a Seção 7.3 contém resultados para as diferentes versões baseadas em AG, já a Seção 7.4 aborda os resultados para o SA, e por fim, a Seção 7.5 efetua uma comparação entre os melhores algoritmos apresentados nesse trabalho.

7.3 Algoritmos Baseados em Algoritmo Genético

Neste trabalho foram utilizados versões do AG com e sem as desigualdades. Os operadores e parâmetros são os seguintes:

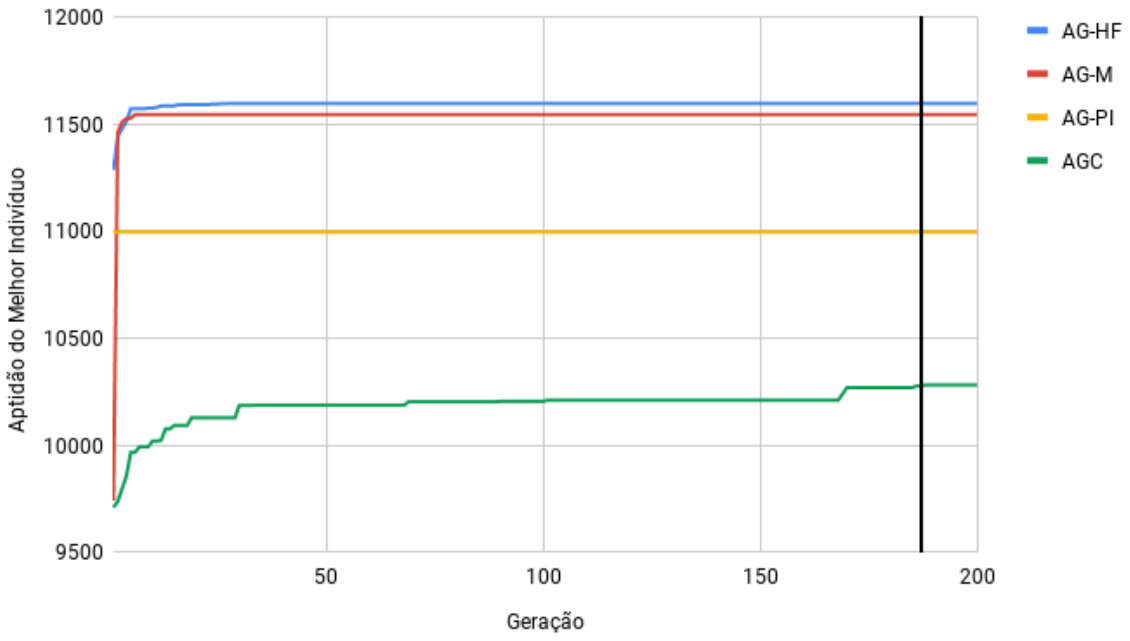
- **Representação:** A representação utilizada é binária, uma vez que as desigualdades válidas são aplicadas em um vetor binário, que representa um indivíduo;
- **População:** O tamanho da população inicial é 50. A cada nova geração, são criados 20 novos indivíduos e como os indivíduos são gerados depende de cada algoritmo;
- **Seleção para Reprodução:** São selecionados 6 indivíduos com uma probabilidade de 30% e então é realizado o torneio. O melhor indivíduo do torneio é selecionado para reprodução;
- **Cruzamento e Mutação:** O Cruzamento utilizado não é uniforme, o individuo com maior aptidão tem uma chance de 60% para seus genes. O resultado no cruzamento é apenas 1 indivíduo. A Mutação é aplicada para cada indivíduo, com exceção do mais apto, com uma probabilidade de 10%;
- **Critérios de Parada:** Foram utilizados dois critérios de parada:
  - *Tempo de execução:* Cada algoritmo foi executado por, no máximo, 1800 segundos (30 minutos).
  - *Repetições:* O algoritmo encerrava sua execução se, em 100 gerações consecutivas, não houver melhora na população.

Os algoritmos serão referenciados segundo os nomes apresentados nas Seções 2.2.4.1, 6.2, 6.3 e 6.4, ou seja, AGC representa a versão do algoritmo genético sem utilizar as desigualdades, *AG-M* é a versão que utiliza as desigualdades como função de mutação, *AG-PI* utiliza as desigualdades na criação da população inicial e por fim o algoritmo *AG-HF* que utiliza as desigualdades em ambas as etapas.

Para um melhor entendimento sobre o comportamento populacional em cada algoritmo, foi efetuada uma análise da variabilidade genética para as instância do conjunto *G*. A análise foi feita levando em consideração as gerações criadas na execução de cada algoritmo e o valor de função objetivo do melhor indivíduo daquela geração. Foi selecionada uma mostragem de 4 instâncias, das 54 pertencentes ao conjunto *G*, que estão representadas nas Figuras 3, 4, 5 e 6. As instâncias utilizadas são respectivamente $G4, G45$, $G24$ e $G50$, com 800, 1000, 2000 e 3000 vértices. Os gráficos contém um linha preta, em posição vertical, indicando que a partir daquela geração nenhum dos algoritmos obteve melhora na população.

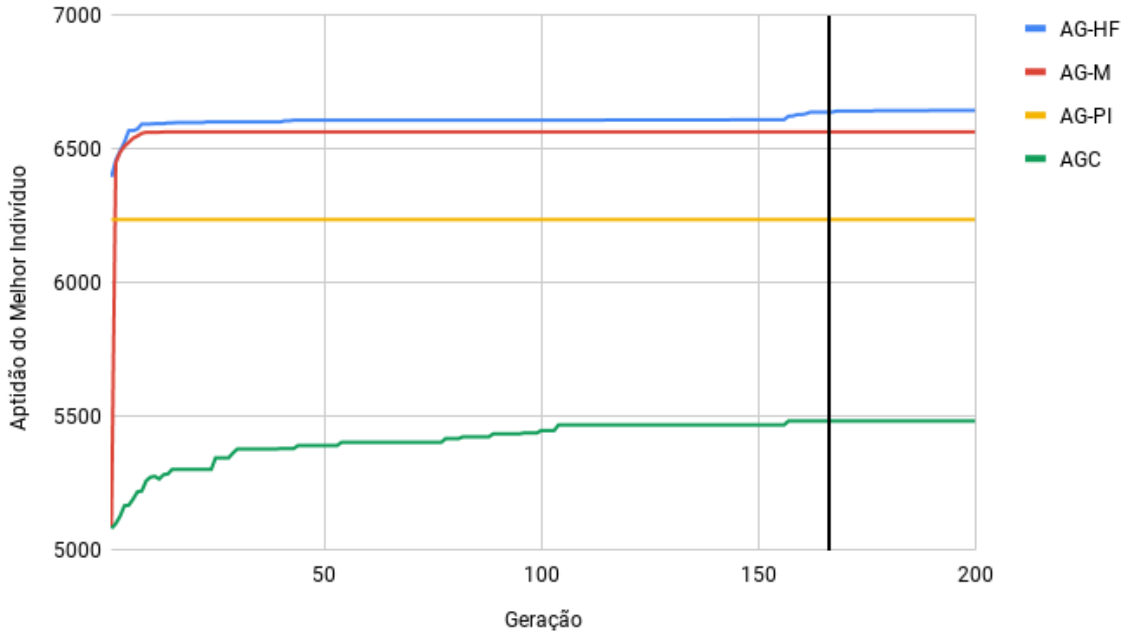
A Tabela 1 apresenta os resultados dos experimentos para as instâncias do tipo *G* e a Tabela 2 apresenta os resultados para as instâncias pertencentes ao conjunto *statistical_physics_application*. Nestas tabelas, está presente a coluna de identificação da instância solucionada (*G, ising, t2gn_seed, t3gn_seed*), o valor da função objetivo para cada meta-heurística (AGC, *AG-PI*, *AG-M*, *AG-HF*) é apresentado, assim o tempo de execução (em segundos) e a quantidade de nós da instância (n).

Figura 3 – Gráfico de Comparação da variabilidade genética da instância *G4*.



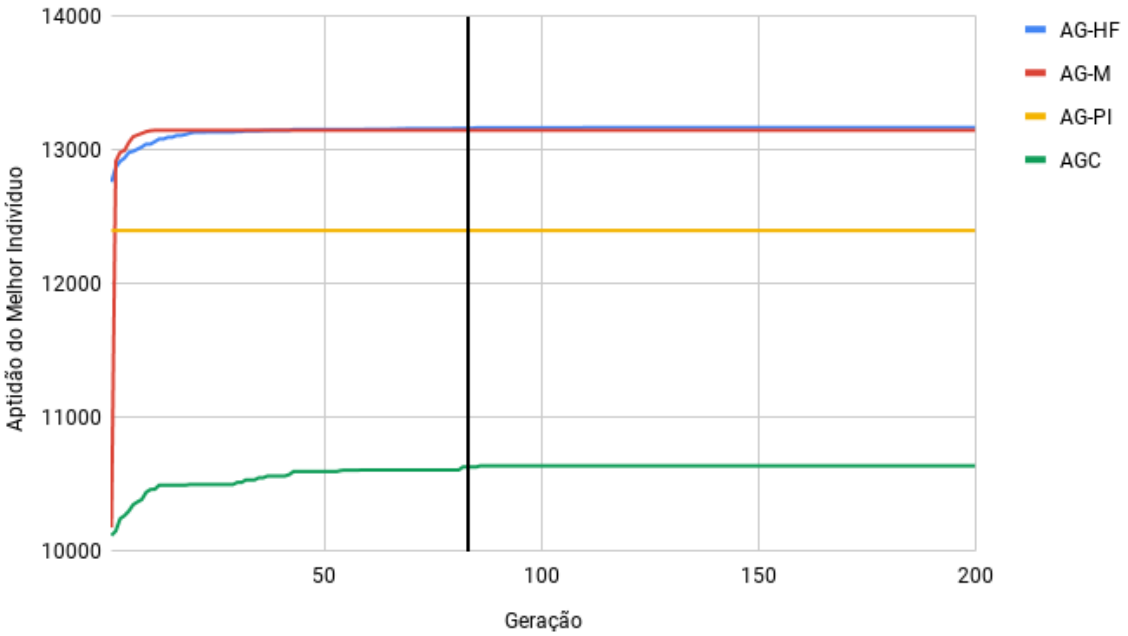
Fonte: Elaborado pelo autor (2018).

Figura 4 – Gráfico de Comparação da variabilidade genética da instância *G45*.



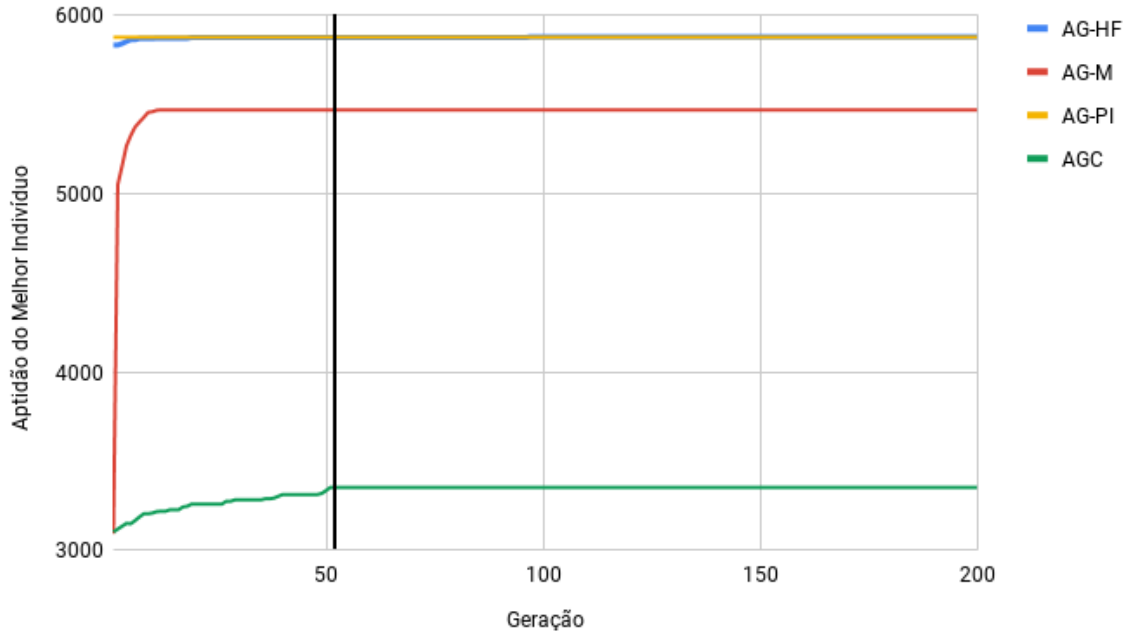
Fonte: Elaborado pelo autor (2018).

Figura 5 – Gráfico de Comparação da variabilidade genética da instância *G24*.



Fonte: Elaborado pelo autor (2018).

Figura 6 – Gráfico de Comparação da variabilidade genética da instância *G50*.



Fonte: Elaborado pelo autor (2018).

Os resultados apresentados na Tabela 1 mostram que em todas as instâncias do conjunto *G_instances*, as heurísticas que utilizaram o conjunto de desigualdades válidas, obtiveram resultados melhores que o AGC. Ainda, para todas as instâncias, é possível afirmar que o *AG-HF* obteve o maior valor de função objetivo, enquanto *AG-PI* foi o algoritmo com o menor tempo de execução.

Mediante análise dos gráficos de variabilidade genética, juntamente com os dados apresentados nas Tabelas 1 e 2, nota-se que o *AG-PI* mantém a solução igual desde o início, ou seja, seu melhor indivíduo é gerado da população inicial e a aplicação dos operadores genéticos do algoritmo não conseguem escapar desse máximo local, fazendo com que o *AG-PI* seja encerrado na condição de parada de 100 iterações sem melhora da solução.

Tabela 1 – Comparação dos GA's para instâncias pertencentes ao conjuntos *G*

		AGC		AG-PI		AG-M		AG-HF	
Instância	n	Valor	Tempo	Valor	Tempo	Valor	Tempo	Valor	Tempo
G1	800	10390	1400,91	11076	**4,89**	11557	43,08	**11609**	163,44
G2	800	10340	682,33	11089	**5,62**	11548	37,87	**11572**	142,2
G3	800	10335	1788,69	11057	**6,21**	11533	42,05	**11566**	153,91
G4	800	10386	1800,87	10988	**5,68**	11617	40,26	**11622**	151,72
G5	800	10343	1727,54	11109	**5,73**	11569	44,88	**11596**	162,81
G6	800	858	1801,42	1559	**5,01**	2136	39,6	**2155**	159,07
G7	800	687	1054,54	1395	**5,36**	1956	37,7	**1985**	156,22
G8	800	702	1124,19	1465	**4,16**	1968	37,3	**1985**	167,35
G9	800	785	1333,31	1479	**5,42**	1994	43,07	**2017**	158,06
G10	800	713	1800,44	1410	**5,15**	1971	37,61	**1985**	142,55
G11	800	244	1281,61	397	**3,38**	518	19,48	**548**	219,05
G12	800	210	768,16	427	**3,38**	510	18,43	**530**	134,08
G13	800	242	1371,83	431	**3,35**	532	17,1	**562**	180,06
G14	800	2670	987,72	2904	**3,7**	3003	20,73	**3045**	271,85
G15	800	2653	1801,54	2862	**3,94**	2988	20,4	**3022**	247,66
G16	800	2654	1345,86	2911	**4**	2995	20,73	**3029**	259,47
G17	800	2648	1179,26	2899	**3,69**	2987	22,92	**3016**	178,55
G18	800	413	906,62	781	**3,66**	928	22,65	**954**	253,52
G19	800	317	1778,55	667	**3,7**	876	25,03	**893**	158,8
G20	800	375	1805,2	705	**4,26**	901	22,11	**918**	170,86
G21	800	386	1086,58	716	**3,36**	882	22,09	**911**	204,6
G22	2000	10733	1817,14	12452	**30,89**	13140	561,95	**13235**	1806,49
G23	2000	10725	1810,15	12348	**31,3**	13179	463,84	**13248**	1804,26
G24	2000	10751	1803,03	12401	**31,23**	13172	463,86	**13219**	1808,87
G25	2000	10724	1816,71	12411	**34,61**	13188	525,7	**13225**	1803,34
G26	2000	10683	1805,57	12378	**28,74**	13134	481,42	**13234**	1802,99
G27	2000	715	1811,88	2299	**26,35**	3163	525,7	**3211**	1808,41
G28	2000	660	1804,18	2341	**33,15**	3099	457,79	**3186**	1806,17
G29	2000	745	1812,36	2361	**36,14**	3221	532,82	**3267**	1802,56
G30	2000	804	1818,42	2354	**31,82**	3248	513,04	**3265**	1806,26
G31	2000	722	1811,01	2309	**26,14**	3135	492,81	**3202**	1809,73
G32	2000	346	1811,02	1022	**21,28**	1294	275,2	**1340**	1804,87
G33	2000	310	1809,63	993	**23,36**	1262	294,06	**1330**	1801,62
G34	2000	314	1806,36	996	**26,24**	1250	242,57	**1334**	1807,67
G35	2000	6374	1812,37	7181	**23,47**	7521	316,66	**7588**	1803,38
G36	2000	6369	1802,77	7120	**23,79**	7508	315,17	**7572**	1804,41
G37	2000	6395	1811,44	7189	**23,76**	7524	303,04	**7618**	1803,97
G38	2000	6386	1810,52	7174	**23,48**	7536	311,98	**7581**	1805,3
G39	2000	602	1813,21	1832	**25,24**	2280	361,17	**2303**	1806,21
G40	2000	524	1811,46	1762	**25,16**	2249	385,72	**2288**	1806,5
G41	2000	597	1807,6	1813	**26,88**	2263	392,09	**2267**	1807,53
G42	2000	596	1808,84	1860	**25,13**	2359	371,13	**2376**	1804,17
G43	1000	5571	1803	6223	**7,69**	6596	65,41	**6621**	299,45
G44	1000	5531	1802,87	6230	**8,49**	6574	63,61	**6611**	492,73
G45	1000	5548	1805,15	6240	**6,64**	6570	58,74	**6609**	639,81
G46	1000	5528	1803,52	6217	**7,64**	6575	62,4	**6588**	274,77
G47	1000	5540	1801,5	6194	**7,49**	6573	67,97	**6618**	370,98
G48	3000	3390	1812,25	5996	**44,47**	5516	976,82	**6000**	1812,58
G49	3000	3380	1808,63	5996	**44,65**	5548	1138,81	**6000**	1815,09
G50	3000	3386	1805,88	5878	**45,54**	5546	1039,22	**5880**	1820,97
G51	1000	3301	1807,44	3653	**6,31**	3778	43,42	**3806**	394,62
G52	1000	3326	1442,96	3651	**5,95**	3789	45,32	**3816**	472,14
G53	1000	3311	1801,39	3632	**5,9**	3789	41,35	**3816**	408,5
G54	1000	3324	1693,41	3633	**5,94**	3781	43,78	**3819**	481,85

Fonte: Elaborado pelo autor (2018).

Tabela 2 – Compração dos GA's para instâncias pertencentes ao conjuntos *statistical_physics_application*

		AGC		AG-PI		AG-M		AG-HF	
Instância	n	Valor	Tempo	Valor	Tempo	Valor	Tempo	Valor	Tempo
ising2.5-100_5555	100	2340918	24,62	2435204	**0,18**	2394925	0,38	**2454597**	1,51
ising2.5-100_6666	100	1905629	14,84	1896585	**0,14**	1955727	0,36	**2031217**	2,15
ising2.5-100_7777	100	3250590	25,91	3110693	**0,21**	3317317	0,38	**3349871**	1,32
ising2.5-150_5555	150	3870774	43	4132167	**0,32**	4178153	0,93	**4346667**	5,69
ising2.5-150_6666	150	3708758	129,18	3662565	**0,24**	3936112	0,91	**4016887**	5,4
ising2.5-150_7777	150	3900185	148,79	3953116	**0,42**	4078810	0,92	**4225139**	8,41
ising2.5-200_5555	200	5266502	71,26	5894973	**0,79**	6067906	1,9	**6231189**	14,01
ising2.5-200_6666	200	5950319	109,17	6021920	**0,5**	6585511	1,82	**6687729**	10,82
ising2.5-200_7777	200	4694669	105,14	4921422	**0,53**	5396342	1,84	**5520288**	12,23
ising2.5-250_5555	250	6308432	184,82	6811246	**0,91**	7632370	3,12	**7797128**	19,96
ising2.5-250_6666	250	5274768	215,93	5463656	**0,84**	6677983	3,15	**6783643**	19,63
ising2.5-250_7777	250	4950083	133,88	5379645	**0,56**	6209497	3,1	**6481287**	25,17
ising2.5-300_5555	300	5946250	161,02	6586701	**0,68**	8308135	5,19	**8453989**	37,15
ising2.5-300_6666	300	6522267	227,16	7263813	**0,72**	8558536	5,23	**8917888**	34,96
ising2.5-300_7777	300	6189018	314,24	6675803	**0,74**	7949141	5	**8167002**	26,68
ising3.0-100_5555	100	2367680	25,96	2394339	**0,13**	2384618	0,37	**2438366**	1,77
ising3.0-100_6666	100	1826815	11,99	1813902	**0,17**	1866707	0,36	**1962241**	1,98
ising3.0-100_7777	100	3193597	15,38	3116876	**0,17**	3271872	0,37	**3321529**	1,7
ising3.0-150_5555	150	3866176	64,37	3951217	**0,36**	4138672	0,92	**4264346**	5,55
ising3.0-150_6666	150	3455334	53,34	3528041	**0,17**	3813185	0,96	**3904437**	5,51
ising3.0-150_7777	150	3718205	23,94	3826922	**0,29**	4065281	0,92	**4204271**	5,86
ising3.0-200_5555	200	5328567	69,72	5692080	**0,61**	5960621	1,99	**6126307**	15,84
ising3.0-200_6666	200	5906252	106,41	5663394	**0,37**	6482809	1,85	**6670323**	10,38
ising3.0-200_7777	200	4714820	69,68	4871023	**0,44**	5336285	1,72	**5524498**	12,28
ising3.0-250_5555	250	6457165	225,95	6804330	**0,86**	7482519	3,24	**7676650**	24,01
ising3.0-250_6666	250	5800462	137,96	6013019	**1,02**	6508469	3,24	**6824960**	30,26
ising3.0-250_7777	250	4883256	130,2	5165481	**0,6**	6047147	2,95	**6267837**	22,7
ising3.0-300_5555	300	5906553	94,69	6859132	**0,72**	8094548	4,85	**8322971**	34,79
ising3.0-300_6666	300	6591818	395,1	7074629	**0,67**	8456092	5,12	**8710318**	24,16
ising3.0-300_7777	300	6049457	309,15	6473743	**0,92**	7858754	4,93	**8074621**	33,42
t2g10_5555	100	5577796	16,93	5757651	**0,13**	5898203	0,36	**6049461**	1,04
t2g10_6666	100	5266708	18,4	5342544	**0,14**	5608107	0,37	**5757868**	1,02
t2g10_7777	100	6056941	13,1	6029957	**0,12**	6458167	0,37	**6509837**	1,37
t2g15_5555	225	11688463	114,01	13294919	**0,74**	14191629	2,31	**14759609**	15,57
t2g15_6666	225	12718916	163,54	13437460	**0,32**	15266460	2,27	**15751559**	19,58
t2g15_7777	225	11932619	131,33	13033790	**0,7**	14513304	2,39	**15095289**	12,05
t2g20_5555	400	13665858	341,98	17927822	**0,87**	23242656	10,06	**24335239**	60,87
t2g20_6666	400	19091367	384,98	22943011	**1,06**	28118028	9,98	**28782027**	75,89
t2g20_7777	400	17843109	311,45	21202391	**0,95**	26947617	10,79	**27930300**	48,88
t3g5_5555	125	10220541	46,8	10524302	**0,24**	10706941	0,58	**10933215**	1,65
t3g5_6666	125	10902474	46,67	10564665	**0,17**	11338141	0,61	**11582216**	1,7
t3g5_7777	125	10894193	96,01	11387917	**0,21**	11488712	0,6	**11552046**	1,93
t3g6_5555	216	12630292	117,56	14886164	**0,7**	16978891	2,24	**17400528**	8,34
t3g6_6666	216	15607938	108,75	17372275	**0,57**	19614147	2,23	**20137679**	9,58
t3g6_7777	216	15320511	160,6	16220131	**0,43**	18861659	2,31	**19388710**	8,22
t3g7_5555	343	17674206	301,23	22165713	**0,82**	27080239	7,56	**27740201**	21,5
t3g7_6666	343	21752406	243,42	26532103	**1,25**	32945704	7,45	**33401146**	31,52
t3g7_7777	343	17618216	245,75	21556595	**0,66**	27838368	7,19	**28675106**	33,26

Fonte: Elaborado pelo autor (2018).

Já no *AG-M* é possível notar que, por iniciar com a população aleatória, este tem o fator de convergência inicial mais alto, ou seja, ele inicia com uma solução próxima da solução do AGC e consegue, nas primeiras gerações, avançar para um solução bem maior, ultrapassando

o *AG-PI* e se aproximando do *AG-HF*. Entretanto, após essa melhora na solução inicial, o *AG-M* também costuma ficar preso em ótimos locais, pelo fato de que as desigualdades são aplicadas a todas as soluções geradas após a função de *Cruzamento*, assim os indivíduos da população começam a ficar com soluções muito semelhantes, prejudicando assim a variabilidade genética do algoritmo.

Como uma alternativa para tirar proveito das vantagens e reduzir os motivos que limitam a eficácia dos algoritmos *AG-PI* e *AG-M*, foi desenvolvido o *AG-HF*, este algoritmo consegue, como é possível ver nos gráficos, partir de uma solução inicial que já é considerada boa e, por manter a função de *Mutação*, do AGC, após a aplicação das desigualdades em cada novo indivíduo gerada, consegue escapar de ótimos locais, ultrapassando assim as soluções do *AG-M*.

Foram selecionamos dois trabalhos da literatura que utilizavam AG, híbridos ou com alterações de estrutura, para resolver o *Max-Cut*. Não foi disponibilizado código fonte ou arquivo executável dos algoritmos e por esse motivo não foi possível comparar tempo de execução para os algoritmos ou aumentar a quantidade de heurísticas selecionadas. O primeiro é o trabalho de (MANSOUR *et al.*, 2006), que utiliza uma abordagem de AGI e é referenciado na Tabela 3 como *IGA*. O segundo trabalho selecionado foi o de (DUARTE *et al.*, 2005), que apresenta uma versão híbrida de AG com VNS e que é referenciada nesse trabalho como *GA-VNS*. Ambos os trabalhos foram selecionados por abordarem o *Max-Cut* com heurísticas e utilizarem instâncias clássicas e disponíveis para *download*.

A Tabela 3 apresenta os resultados dos experimentos para as instâncias do tipo *G*. Nesta tabela, está presente a coluna de identificação da instância solucionada ($G1, \ldots, G54$), assim como o valor da função objetivo para cada algoritmo (*AG-HF*, *IGA*, *GA-VNS*) e a quantidade de nós da instância (n). O símbolo $-$, presente em algumas linhas da tabela, significa que o algoritmo em questão não apresenta resultado para essa instância e os valores que estão em negrito, na coluna de valor, significam que o algoritmo obteve um resultado melhor ou igual aos demais resultados exibidos para a instância.

Tabela 3 – Comparação dos algoritmos *AG-HF, IGA* e *GA-VNS* utilizando as instâncias do conjunto *G*

		AG-HF	IGA	GA-VNS
Instância	n	Valor	Valor	Valor
G1	800	11609	11411,9	**11624**
G2	800	11572	11417,1	**11620**
G3	800	11566	11435,3	**11622**
G4	800	**11622**	11442,4	-
G5	800	**11596**	11450,1	-
G6	800	2155	-	-
G7	800	1985	-	-
G8	800	1985	-	-
G9	800	2017	-	-
G10	800	1971	-	-
G11	800	548	-	**562**
G12	800	530	-	**554**
G13	800	562	-	**580**
G14	800	3045	2998,08	**3061**
G15	800	3022	2979,85	**3046**
G16	800	3029	2983,94	**3047**
G17	800	**3016**	2980,29	-
G18	800	954	-	-
G19	800	893	-	-
G20	800	918	-	-
G21	800	911	-	-
G22	2000	13235	-	**13318**
G23	2000	13248	-	**13322**
G24	2000	13219	-	**13319**
G25	2000	13225	-	-
G26	2000	13234	-	-
G27	2000	3211	-	-
G28	2000	3186	-	-
G29	2000	3267	-	-
G30	2000	3265	-	-
G31	2000	3202	-	-
G32	2000	1340	-	**1392**
G33	2000	1330	-	**1362**
G34	2000	1334	-	**1368**
G35	2000	7588	-	**7665**
G36	2000	7572	-	**7643**
G37	2000	7618	-	**7657**
G38	2000	7581	-	-
G39	2000	2303	-	-
G40	2000	2288	-	-
G41	2000	2267	-	-
G42	2000	2376	-	-
G43	1000	6621	6474,56	**6655**
G44	1000	6611	6470,49	**6649**
G45	1000	6609	6469,92	**6634**
G46	1000	**6588**	6475,64	-
G47	1000	**6618**	6475,07	-
G48	3000	**6000**	-	6000
G49	3000	**6000**	-	6000
G50	3000	**5880**	-	5880
G51	1000	**3806**	3757,44	-
G52	1000	**3816**	3761,92	-
G53	1000	**3816**	3758,96	-
G54	1000	**3819**	3758,62	-

Fonte: Elaborado pelo autor (2018).

Os resultados presentes na Tabela 3 mostram que o algoritmo detentor do maior valor de função objetivo nas 9, de 54, instâncias onde as 3 heurísticas apresentam resultados, foi o *AG-VNS*. É possível ressaltar que a obtenção de maiores resultados pelo algoritmo *AG-VNS* se dá pelo fato que a estratégia de busca híbrida utilizando VNS consegue atingir soluções mais diversificadas, em contraste com o codificação do corte de otimalidade, que sempre aproxima a solução de um determinado subconjunto de soluções. Levando para comparação apenas os algoritmos *AG-HF* e *GA-VNS*, podemos ver que, em apenas 3, de 24, instâncias onde ambos apresentam resultados, os valores são iguais, nos demais casos o algoritmo *GA-VNS* foi melhor. Já efetuando a comparação apenas entre *AG-HF* e *IGA*, nas 18 instâncias onde ambos os algoritmos apresentam resultados o algoritmo *AG-HF* obteve melhores resultados em todas.

7.4 Algoritmo Baseado em Têmpera Simulada

Esta Seção apresenta os resultados obtidos com o SA com e sem as desigualdades válidas, assim como apresentado na Seção 6.5, a versão sem desigualdades será referenciada como *SA-MYK* (MYK - Por conta do autor), enquanto a versão que utiliza as desigualdades está intitulada de *SA-H*(H - Híbrido). Os operadores e parâmetros utilizados são os seguintes:

- **Representação:** Houve a adição da representação da solução por meio de um vetor binário, 0 e 1 no *SA-H*;
- **Uso das Desigualdades:** *SA-H* modifica a solução utilizando as desigualdades a cada 50 iterações onde não foram aceitas modificações na solução;
- **Temperatura:** Em ambos os algoritmos foram utilizadas duas variáveis associadas a temperatura, a *Temperatura Máxima* (T_{max}) foi de 10000 e a variável *calor* inicia com 0 e aumenta $2e^{-6}$ a cada iteração.
- **Critérios de Parada:**
  - $calor \geq t_{max}$*:* Cada algoritmos poderia executar até que a variável *calor* atingisse o valor 10000, ou seja, valor de t_{max};
  - *Tempo de execução:* Cada algoritmo foi executado por, no máximo, 1800 segundos (30 minutos).

A Tabela 4 apresenta os resultados dos experimentos para as instâncias do tipo G e a Tabela 5 apresenta os resultados para as instâncias pertencentes ao conjunto *statistical_physics_application*. Nestas tabelas, está presente a coluna de identificação da instância solucionada (G_i*, ising, t2gn_seed, t3gn_seed*), o valor da função objetivo para cada meta-

heurística (SA-MYK, SA-H), assim como o tempo de execução (em segundos), a quantidade de nós da instância (n) e os valores em negrito indicam qual algoritmo obteve melhor resultado, seja menor tempo de execução ou maior valor de FO.

Os resultados apresentados nas Tabelas 4 e 5 mostram que a utilização das desigualdades para a meta-heurística SA gera uma diferença menor de resultados, em relação com as comparações apresentadas na seção 7.3, entre as diferentes versões do AG. Na Tabela 4 é possível notar que em 49, das 54 instâncias do conjunto *G*, o SA-MYK obteve um menor tempo de execução, nas outras 5 instâncias o tempo de execução foi igual. Já para o valor de função objetivo é possível destacar que em 25, das 54 instâncias, o SA-H obteve um valor maior, ou igual, ao valor obtido por SA-MYK. Já para os resultados apresentados na Tabela 5, em nenhuma das 48 instâncias o algoritmo SA-H obteve um menor tempo de execução e conseguiu um valor de função objetivo maior ou igual ao algoritmo SA-MIK em apenas 7, das 48 instâncias.

Mediante análise dos resultados obtidos para as diferentes versões do SA que foram implementadas, é possível notar que houve uma melhora na eficiência da versão do SA utilizando as desigualdades, mas houve a necessidade de um tempo maior de execução, por conta da quantidade de vezes em que as desigualdades são aplicadas à solução, esse tempo é reduzido de acordo com o acréscimo do valor de uso das desigualdades. Já na questão do valor de função objetivo, as desigualdades se mostraram mais eficientes para as instâncias com maiores espaços de solução, uma vez que o ponto forte do SA está em percorrer soluções vizinhas com probabilidade de aceitação de soluções consideradas "piores"em busca de fugir de ótimos locais, o que faz com que não seja possível tirar grande proveito das desigualdades, já que elas podem resultar no mesmo vetor de solução, dependendo da semelhança dos vetores de entrada entrada.

Tabela 4 – Comparação dos SA's para as instâncias pertencentes ao conjunto *G*

		SA-MYK		SA-H	
Instância	n	Valor	Tempo	Valor	Tempo
G1	800	11582	1800	**11624**	1800
G2	800	11570	1800	**11618**	1800
G3	800	11555	1800	**11622**	1800
G4	800	11571	1800	**11641**	1800
G5	800	11556	1800	**11631**	1800
G6	800	2178	**1330,54**	**2178**	1800
G7	800	2006	**1258,59**	**2006**	1800
G8	800	1999	**1314,81**	**2005**	1800
G9	800	**2054**	**1075,36**	2050	1800
G10	800	1999	**1167,95**	**2000**	1800
G11	800	**564**	**1078,06**	546	1800
G12	800	**556**	**1005,99**	540	1800
G13	800	**582**	**1056,72**	570	1800
G14	800	**3062**	**1084,09**	3048	1800
G15	800	**3049**	**1085,87**	3036	1800
G16	800	**3052**	**1332,15**	3036	1800
G17	800	**3045**	**1147,79**	3030	1800
G18	800	**990**	**923,14**	986	1800
G19	800	**904**	**1032,15**	900	1800
G20	800	**941**	**969,53**	937	1800
G21	800	**927**	**1022,72**	923	1800
G22	2000	13150	**1796,04**	**13348**	1800
G23	2000	13111	**1755,37**	**13316**	1800
G24	2000	13135	**1687,64**	**13325**	1800
G25	2000	13105	**1710,06**	**13327**	1800
G26	2000	13093	**1767,69**	**13307**	1800
G27	2000	**3341**	**1082,47**	3328	1800
G28	2000	**3298**	**1320,07**	3292	1800
G29	2000	3394	**1118,3**	**3401**	1800
G30	2000	**3412**	**1152,01**	3410	1800
G31	2000	3302	**1065,08**	**3306**	1800
G32	2000	**1408**	**1019,34**	1366	1800
G33	2000	**1380**	**1016,42**	1334	1800
G34	2000	**1384**	**966,95**	1346	1800
G35	2000	7489	**1551,6**	**7567**	1800
G36	2000	7470	**1360,41**	**7549**	1800
G37	2000	7484	**1356,64**	**7554**	1800
G38	2000	7478	**1353,87**	**7555**	1800
G39	2000	**2404**	**1229,96**	2388	1800
G40	2000	**2397**	**1001,97**	2380	1800
G41	2000	**2394**	**1232,79**	2377	1800
G42	2000	**2466**	**1081,77**	2448	1800
G43	1000	**6659**	**1483,87**	6652	1800
G44	1000	6647	**1366,53**	**6647**	1800
G45	1000	**6653**	**1346,99**	6650	1800
G46	1000	**6646**	**1326,12**	6645	1800
G47	1000	6651	**1405,67**	**6654**	1800
G48	3000	6000	**1029,18**	**6000**	1800
G49	3000	6000	**1219,39**	**6000**	1800
G50	3000	5858	**1105,66**	**5870**	1800
G51	1000	**3841**	**1384,42**	3830	1800
G52	1000	**3844**	**1174,01**	3829	1800
G53	1000	**3842**	**1254,81**	3828	1800
G54	1000	**3845**	**1392,39**	3835	1800

Fonte: Elaborado pelo autor (2018).

Tabela 5 – Comparação dos SA's para as instâncias pertencentes ao conjunto *statistical_physics_application*

		SA-MYK		SA-H	
Instância	n	Valor	Tempo	Valor	Tempo
ising2.5-100_5555	100	**2460049**	**941,46**	2445100	1800
ising2.5-100_6666	100	2015483	**1153,42**	**2031217**	1800
ising2.5-100_7777	100	**3363230**	**934,16**	3323738	1800
ising2.5-150_5555	150	**4356439**	**1017,67**	4278472	1800
ising2.5-150_6666	150	**4043795**	**1092,06**	3960879	1800
ising2.5-150_7777	150	**4232158**	**1090,51**	4158088	1800
ising2.5-200_5555	200	**6276406**	**1313,36**	6107625	1800
ising2.5-200_6666	200	**6748754**	**1131,19**	6605210	1800
ising2.5-200_7777	200	**5550304**	**1186,36**	5454774	1800
ising2.5-250_5555	250	**7879896**	**1411,9**	7623729	1800
ising2.5-250_6666	250	**6885523**	**1498,23**	6666151	1800
ising2.5-250_7777	250	**6559859**	**1468,07**	6309936	1800
ising2.5-300_5555	300	**8525055**	**1430,88**	8184414	1800
ising2.5-300_6666	300	**9047021**	**1470,76**	8645119	1800
ising2.5-300_7777	300	**8237242**	**1630,46**	7981607	1800
ising3.0-100_5555	100	**2443375**	**1032,6**	2426205	1800
ising3.0-100_6666	100	**1979275**	**1097,84**	1967148	1800
ising3.0-100_7777	100	**3335814**	**942,46**	3293044	1800
ising3.0-150_5555	150	**4266248**	**1032,91**	4201668	1800
ising3.0-150_6666	150	**3944934**	**1243,32**	3874369	1800
ising3.0-150_7777	150	**4193251**	**1147,94**	4114095	1800
ising3.0-200_5555	200	**6193441**	**1294,19**	5979828	1800
ising3.0-200_6666	200	**6723291**	**1250,54**	6562392	1800
ising3.0-200_7777	200	**5532558**	**1203,8**	5390905	1800
ising3.0-250_5555	250	**7785737**	**1504,32**	7568790	1800
ising3.0-250_6666	250	**6862048**	**1411,34**	6538102	1800
ising3.0-250_7777	250	**6406856**	**1273,66**	6097172	1800
ising3.0-300_5555	300	**8413058**	**1349,53**	8120010	1800
ising3.0-300_6666	300	**8865294**	**1468,42**	8436151	1800
ising3.0-300_7777	300	**8189561**	**1496,12**	7897057	1800
t2g10_5555	100	6049461	**913,37**	**6049461**	1800
t2g10_6666	100	5757868	**939,95**	**5757868**	1800
t2g10_7777	100	6509837	**898,73**	**6509837**	1800
t2g15_5555	225	**15051133**	**1047,78**	14617991	1800
t2g15_6666	225	**15752637**	**920,03**	15511945	1800
t2g15_7777	225	**15266618**	**875,95**	14756399	1800
t2g20_5555	400	**24838942**	**895,06**	23196444	1800
t2g20_6666	400	**29201142**	**847,98**	27805630	1800
t2g20_7777	400	**28140879**	**844,14**	26768741	1800
t3g5_5555	125	10933215	**1025,7**	**10933215**	1800
t3g5_6666	125	11582216	**920,6**	**11582216**	1800
t3g5_7777	125	11552046	**931**	**11552046**	1800
t3g6_5555	216	**17434469**	**865,73**	17357414	1800
t3g6_6666	216	**20217380**	**921,42**	19914721	1800
t3g6_7777	216	**19475011**	**1109,17**	19226377	1800
t3g7_5555	343	**28302918**	**899,27**	27318360	1800
t3g7_6666	343	**33611981**	**889,23**	32635220	1800
t3g7_7777	343	**29110043**	**847,93**	27971022	1800

Fonte: Elaborado pelo autor (2018).

7.5 Algoritmo Genético e Têmpera Simulada

Esta seção apresenta um comparativo entre os algoritmos desenvolvidos neste trabalho. A análise foi realizada utilizando a melhor versão baseada em Algoritmo Genético, AG-HF, e a versão do Têmpera Simulada que utiliza as desigualdades, SA-H. Na tabela 6 são apresentados os resultados dos experimentos para as instâncias do tipo *G*, enquanto na Tabela 7 estão os resultados para as instâncias pertencentes ao conjunto *statistical_physics_application*. Nestas tabelas está presente a coluna de identificação da instância solucionada (G_i*, ising, t2gn_seed, t3gn_seed*), o valor da função objetivo para cada meta-heurística (AG-HF, SA-H), assim como o tempo de execução (em segundos), a quantidade de nós da instância (n) e os valores em negrito indicando qual algoritmo obteve melhor resultado, seja menor tempo de execução ou maior valor de FO.

Os resultados apresentados nas Tabelas 6 e 7 dão maior certeza as conclusões obtidas em (DUNNING *et al.*, 2018), uma vez que, para as instâncias do tipo *G*, com maior número de vértices, o algoritmo SA-H obteve os maiores valores de função objetivo em 46, das 54 instâncias, e no quesito de tempo, o algoritmo GA-HF obteve um tempo de execução menor em 13, das 54 instâncias, nas demais 41 instâncias o tempo de execução foi o mesmo para ambos os algoritmos. Já os resultados da Tabela 7, contendo as instâncias pertencentes ao conjunto *statistical_physics_application*, ressaltam que em 40, das 48 instâncias, o algoritmo AG-HF obteve os maiores valores de função objetivo, nas outras 8 instâncias os valores obtidos foram iguais. AG-F também obteve o menor tempo de execução para todas as instâncias.

Tabela 6 – Comparação dos algoritmos SA-H e AG-HF para as instâncias pertencentes ao conjunto *G*

		AG-HF		SA-H	
Instância	n	Valor	Tempo	Valor	Tempo
G1	800	11609	**163,44**	**11624**	1800
G2	800	11572	**142,2**	**11618**	1800
G3	800	11566	**153,91**	**11622**	1800
G4	800	11622	**151,72**	**11641**	1800
G5	800	11596	**162,81**	**11631**	1800
G6	800	2155	**159,07**	**2178**	1800
G7	800	1985	**156,22**	**2006**	1800
G8	800	1985	**167,35**	**2005**	1800
G9	800	2017	**158,06**	**2050**	1800
G10	800	1971	**142,55**	**2000**	1800
G11	800	**548**	**219,05**	546	1800
G12	800	530	**134,08**	**540**	1800
G13	800	562	**180,06**	**570**	1800
G14	800	3045	**271,85**	**3048**	1800
G15	800	3022	**247,66**	**3036**	1800
G16	800	3029	**259,47**	**3036**	1800
G17	800	3016	**178,55**	**3030**	1800
G18	800	954	**253,52**	**986**	1800
G19	800	893	**158,8**	**900**	1800
G20	800	918	**170,86**	**937**	1800
G21	800	911	**204,6**	**923**	1800
G22	2000	13235	1800	**13348**	1800
G23	2000	13248	1800	**13316**	1800
G24	2000	13219	1800	**13325**	1800
G25	2000	13225	1800	**13327**	1800
G26	2000	13234	1800	**13307**	1800
G27	2000	3211	1800	**3328**	1800
G28	2000	3186	1800	**3292**	1800
G29	2000	3267	1800	**3401**	1800
G30	2000	3265	1800	**3410**	1800
G31	2000	3202	1800	**3306**	1800
G32	2000	1340	1800	**1366**	1800
G33	2000	1330	1800	**1334**	1800
G34	2000	1334	1800	**1346**	1800
G35	2000	**7588**	1800	7567	1800
G36	2000	**7572**	1800	7549	1800
G37	2000	**7618**	1800	7554	1800
G38	2000	**7581**	1800	7555	1800
G39	2000	2303	1800	**2388**	1800
G40	2000	2288	1800	**2380**	1800
G41	2000	2267	1800	**2377**	1800
G42	2000	2376	1800	**2448**	1800
G43	1000	6621	**299,45**	6652	1800
G44	1000	6611	**492,73**	6647	1800
G45	1000	6609	**639,81**	6650	1800
G46	1000	6588	**274,77**	6645	1800
G47	1000	6618	**370,98**	6654	1800
G48	3000	6000	1800	6000	1800
G49	3000	6000	1800	6000	1800
G50	3000	**5880**	1800	5870	1800
G51	1000	3806	**394,62**	**3830**	1800
G52	1000	3816	**472,14**	**3829**	1800
G53	1000	3816	**408,5**	**3828**	1800
G54	1000	3819	**481,85**	**3835**	1800

Fonte: Elaborado pelo autor (2018).

Tabela 7 – Comparação dos algoritmos SA-H e AG-HF para as instâncias pertencentes ao conjunto *statistical_physics_application*

		AG-HF		SA-H	
Instância	n	Valor	Tempo	Valor	Tempo
ising2.5-100_5555	100	**2454597**	**1,51**	2445100	1800
ising2.5-100_6666	100	2031217	**2,15**	2031217	1800
ising2.5-100_7777	100	**3349871**	**1,32**	3323738	1800
ising2.5-150_5555	150	**4346667**	**5,69**	4278472	1800
ising2.5-150_6666	150	**4016887**	**5,4**	3960879	1800
ising2.5-150_7777	150	**4225139**	**8,41**	4158088	1800
ising2.5-200_5555	200	**6231189**	**14,01**	6107625	1800
ising2.5-200_6666	200	**6687729**	**10,82**	6605210	1800
ising2.5-200_7777	200	**5520288**	**12,23**	5454774	1800
ising2.5-250_5555	250	**7797128**	**19,96**	7623729	1800
ising2.5-250_6666	250	**6783643**	**19,63**	6666151	1800
ising2.5-250_7777	250	**6481287**	**25,17**	6309936	1800
ising2.5-300_5555	300	**8453989**	**37,15**	8184414	1800
ising2.5-300_6666	300	**8917888**	**34,96**	8645119	1800
ising2.5-300_7777	300	**8167002**	**26,68**	7981607	1800
ising3.0-100_5555	100	**2438366**	**1,77**	2426205	1800
ising3.0-100_6666	100	1962241	**1,98**	1967148	1800
ising3.0-100_7777	100	**3321529**	**1,7**	3293044	1800
ising3.0-150_5555	150	**4264346**	**5,55**	4201668	1800
ising3.0-150_6666	150	**3904437**	**5,51**	3874369	1800
ising3.0-150_7777	150	**4204271**	**5,86**	4114095	1800
ising3.0-200_5555	200	**6126307**	**15,84**	5979828	1800
ising3.0-200_6666	200	**6670323**	**10,38**	6562392	1800
ising3.0-200_7777	200	**5524498**	**12,28**	5390905	1800
ising3.0-250_5555	250	**7676650**	**24,01**	7568790	1800
ising3.0-250_6666	250	**6824960**	**30,26**	6538102	1800
ising3.0-250_7777	250	**6267837**	**22,7**	6097172	1800
ising3.0-300_5555	300	**8322971**	**34,79**	8120010	1800
ising3.0-300_6666	300	**8710318**	**24,16**	8436151	1800
ising3.0-300_7777	300	**8074621**	**33,42**	7897057	1800
t2g10_5555	100	6049461	**1,04**	6049461	1800
t2g10_6666	100	5757868	**1,02**	5757868	1800
t2g10_7777	100	6509837	**1,37**	6509837	1800
t2g15_5555	225	**14759609**	**15,57**	14617991	1800
t2g15_6666	225	**15751559**	**19,58**	15511945	1800
t2g15_7777	225	**15095289**	**12,05**	14756399	1800
t2g20_5555	400	**24335239**	**60,87**	23196444	1800
t2g20_6666	400	**28782027**	**75,89**	27805630	1800
t2g20_7777	400	**27930300**	**48,88**	26768741	1800
t3g5_5555	125	10933215	**1,65**	10933215	1800
t3g5_6666	125	11582216	**1,7**	11582216	1800
t3g5_7777	125	11552046	**1,93**	11552046	1800
t3g6_5555	216	**17400528**	**8,34**	17357414	1800
t3g6_6666	216	**20137679**	**9,58**	19914721	1800
t3g6_7777	216	**19388710**	**8,22**	19226377	1800
t3g7_5555	343	**27740201**	**21,5**	27318360	1800
t3g7_6666	343	**33401146**	**31,52**	32635220	1800
t3g7_7777	343	**28675106**	**33,26**	27971022	1800

Fonte: Elaborado pelo autor (2018).

8 CONCLUSÃO

Neste trabalho 4 meta-heurísticas baseadas em Algoritmo Genético e Têmpera Simulada, que utilizam um conjunto de desigualdades válidas baseadas em condições de otimalidade em sua estrutura, foram apresentadas juntamente com uma análise dos seus desempenhos em comparação com os algoritmos da literatura e as respectivas versões das meta-heurísticas sem a utilização das desigualdades. A análise foi realizada considerando o tempo de execução e o valor de função objetivo obtido por cada algoritmo. Para tal, foi utilizada uma amostra de 102 instâncias pertencentes ao conjuntos *G* e *statistical_physics_application*. Embora os experimentos para os algoritmos baseados em AG considerem um tempo de execução limitado a 1800 segundos e uma quantidade máxima de 100 gerações sem melhoria da população, enquanto os experimentos para os algoritmos baseados em SA foi considerado um tempo de execução limitado também a 1800 segundos e um valor de temperatura máxima de 10000 para cada execução, é possível afirmar que o aumento dos valores dessas condições não afetaria, com grande diferença, o resultado final do algoritmo.

Sendo assim, é possível concluir que as versões do AG que utilizam as desigualdades válidas, propostas neste trabalho, podem ser consideradas superiores ao algoritmo genético clássico, principalmente no requisito de valor de função objetivo, mas não foi possível superar a melhor versão baseada em AG presente na literatura para o *Max-Cut*, no melhor dos casos para a melhor versão baseada em AG deste trabalho, *AG-HF*, foi possível ser competitivo. Para a versão do algoritmo baseado em SA, a utilização do conjunto de desigualdades válidas fez com que o algoritmo atingisse melhores resultados em algumas instâncias de teste, superando assim a melhor versão apresentada por algoritmos baseados em SA para o *Max-Cut*, mas o impacto da utilização das desigualdades válidas, para todas as instâncias de teste, não foi tão grande quanto a diferença obtida nas versões dos algoritmos baseados em AG.

8.1 Trabalhos Futuros

Quanto a utilização de algoritmos heurísticos e híbridos para o *Max-Cut*, embora a utilização das desigualdades torne os algoritmos melhores que as versões sem as desigualdades, para instâncias utilizadas, os algoritmos propostos conseguem, na maioria dos casos, ser competitivos com o estado da arte do *Max-Cut*. Espera-se em trabalhos futuros explorar de melhor forma as desigualdades provadas, aplicando-as em outras meta-heurísticas, tais como

Algoritmo Genético de Chaves Aleatórias, Busca Dispersa, Busca em Tabu etc, de maneira que seja utilizado com maior proveito possível das modificações que são efetuadas na solução. Outra possível proposta seria o aperfeiçoamento das propriedades do conjunto de desigualdades, buscando maneiras mais eficientes e rápidas de aplicar-las as soluções em questão.

REFERÊNCIAS

ANJOS, M. F. **New convex relaxations for the maximum cut and VLSI layout problems**. Tese (Doutorado), 2001.

BARAHONA, F. The max-cut problem on graphs not contractible to k5. **Operations Research Letters**, North-Holland, v. 2, n. 3, p. 107–111, 1983.

BARAHONA, F.; GRÖTSCHEL, M.; JÜNGER, M.; REINELT, G. An application of combinatorial optimization to statistical physics and circuit layout design. **Operations Research**, INFORMS, v. 36, n. 3, p. 493–513, 1988.

BOROS, E.; HAMMER, P. L. The max-cut problem and quadratic 0–1 optimization; polyhedral aspects, relaxations and bounds. **Annals of Operations Research**, Springer, v. 33, n. 3, p. 151–180, 1991.

BOYKOV, Y. Y.; JOLLY, M.-P. Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In: IEEE. **Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on**. [S.l.], 2001. v. 1, p. 105–112.

BURER, S.; MONTEIRO, R.; ZHANG, Y. Rank-two relaxation heuristic for max-cut and other binary quadratic problems. **Siam Journal on Optimization**, v. 12, n. 2, p. 503–521, 2001/2002.

CAIXETA-FILHO, J. V. **Pesquisa Operacional**. São Paulo: Atlas, 2001.

DANTZIG, G. B. **Computational algorithm of the revised simplex method**. [S.l.]: Rand Corporation, 1953.

DEZA, M.; LAURENT, M. **Geometry of Cuts and Metrics: Algotithms and Combinatorics**. Berlin: Springer, 1997.

DUARTE, A.; SÁNCHEZ, Á.; FERNÁNDEZ, F.; CABIDO, R. A low-level hybridization between memetic algorithm and vns for the max-cut problem. In: ACM. **Proceedings of the 7th annual conference on Genetic and evolutionary computation**. [S.l.], 2005. p. 999–1006.

DUNNING, I.; GUPTA, S.; SILBERHOLZ, J. What works best when? a systematic evaluation of heuristics for max-cut and qubo. **INFORMS Journal on Computing**, 2018.

FESTA, P.; PARDALOS, P.; RESENDE, M.; RIBEIRO, C. Grasp and vns for max-cut. In: **Extended Abstracts of the Fourth Metaheuristics International Conference**. [S.l.: s.n.], 2001. p. 371–376.

FESTA, P.; PARDALOS, P. M.; RESENDE, M. G.; RIBEIRO, C. C. Randomized heuristics for the max-cut problem. **Optimization methods and software**, Taylor & Francis, v. 17, n. 6, p. 1033–1058, 2002.

GARCIA, S.; GUERREIRO, R.; CORRAR, L. J. Teoria das restrições e programação linear. In: **Congresso Internacional de Costos**. [S.l.: s.n.], 1997.

GOEMANS, M. X.; WILLIAMSON, D. P. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. **J. ACM**, v. 42, n. 6, p. 1115–1145, Nov 1995.

GOEMANS, M. X.; WILLIAMSON, D. P. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. **Journal of the ACM (JACM)**, ACM, v. 42, n. 6, p. 1115–1145, 1995.

HAMER, P. The max-cut problem and quadratic 0-1 optimization; polyhedral aspects, relaxations and bounds. **Annals of Operations Research**, v. 33, n. 3, p. 151–180, 1991.

HARTMANN, A. K.; RIEGER, H. **New optimization algorithms in physics**. [S.l.]: John Wiley & Sons, 2006.

HELMBERG, C.; RENDL, F. A spectral bundle method for semidefinite programming. **SIAM Journal on Optimization**, SIAM, v. 10, n. 3, p. 673–696, 2000.

HELMBERG, C.; RENDL, F.; VANDERBEI, R.; WOLKOWICZ, H. An interior-ponit method for semidefinite programming. **Siam Journal on Optimization**, v. 6, n. 2, p. 342–361, 1996.

HILLIER, F. S.; LIEBERMAN, G. J. **Introdução à pesquisa operacional**. [S.l.]: McGraw Hill Brasil, 2013.

KARP, R. M. **Reducibility among Combinatorial Problems**. [S.l.]: Springer US, 1972.

KIM, S.-H.; KIM, Y.-H.; MOON, B.-R. A hybrid genetic algorithm for the max cut problem. In: MORGAN KAUFMANN PUBLISHERS INC. **Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation**. [S.l.], 2001. p. 416–423.

KOCHENBERGER, G. A.; HAO, J.-K.; LÜ, Z.; WANG, H.; GLOVER, F. Solving large scale max cut problems via tabu search. **Journal of Heuristics**, Springer, v. 19, n. 4, p. 565–571, 2013.

KRISLOCK, N.; MALICK, J.; ROUOIN, F. Improved semidefinite bounding procedure for solving max-cut problems to optimality. **Mathematical Programming**, v. 143, n. 1–2, p. 61–86, 2014.

LAURENT, M.; POLJAK, S.; RENDL, F. Connections between semidefinite relaxations of the max-cut and stable set problems. **Mathematical Programming**, v. 77, n. 1, p. 225–246, Apr 1997. ISSN 1436-4646.

LIERS, F. **Contributions to determining exact ground-states of Ising spin-glasses and to their physics**. Tese (Doutorado) — Universität zu Köln, 2004.

LIERS, F.; JÜNGER, M.; REINELT, G.; RINALDI, G. Computing exact ground states of hard ising spin glass problems by branch-and-cut. **New optimization algorithms in physics**, Berlin: Wiley VCH, v. 50, n. 47-68, p. 6, 2004.

LIERS, F.; NIEBERG, T.; PARDELLA, G. Via minimization in vlsi chip design-application of a planar max-cut algorithm. 2011.

LUNA, H. P.; GOLDBARG, M. C. Otimização combinatória e programação linear. **Rio de Janeiro: Campus**, 2000.

MANSOUR, N.; AWAD, M.; EL-FAKIH, K. Incremental genetic algorithm. **The International Arab Journal of Information Technology**, v. 3, n. 1, p. 42–47, 2006.

MITCHELL, M. **An introduction to genetic algorithms**. [S.l.]: MIT press, 1998.

MYKLEBUST, T. G. Solving maximum cut problems by simulated annealing. **arXiv preprint arXiv:1505.03068**, 2015.

POLJAK, S.; TUZA, Z. **The Max-Cut problem - a survey**. Nankang, Taipei: Academica Sinica, 1993. 91 p. Disponível em: <http://eprints.sztaki.hu/256/>.

PUCCINI, A. d. L. **Introdução à Programação Linear**. Rio de Janeiro: LTC, 1980.

RENDL, F.; RINALDI, G.; WIEGELE, A. A branch and bound algorithm for max-cut based on combining semidefinite and polyhedral relaxations. In: SPRINGER. **International Conference on Integer Programming and Combinatorial Optimization**. [S.l.], 2007. p. 295–309.

SIMONE, C. D.; DIEHL, M.; JÜNGER, M.; MUTZEL, P.; REINELT, G.; RINALDI, G. Exact ground states of ising spin glasses: New experimental results with a branch-and-cut algorithm. **Journal of Statistical Physics**, Springer, v. 80, n. 1-2, p. 487–496, 1995.

SIMONE, C. D.; RINALDI, G. A cutting plane algorithm for the max-cut problem. **Optimization Methods and Software**, Taylor & Francis, v. 3, n. 1-3, p. 195–214, 1994.

SOARES, P.; CAMPÊLO, M.; RODRIGUES, C. D.; MICHELON, P. t-linearizalização de funções quadráticas de variáveis binárias. **Anais do XLIX SBPO**, p. 2569–2580, 2017.

SOARES, P. L. B. **Problemas quadráticos binários: abordagem teórica e computacional**. Tese — Universidade Federal do Ceará, Fortaleza, CE, nov 2018.

SOUSA, S. de; HAXHIMUSA, Y.; KROPATSCH, W. G. Estimation of distribution algorithm for the max-cut problem. In: SPRINGER. **International Workshop on Graph-Based Representations in Pattern Recognition**. [S.l.], 2013. p. 244–253.

STEVEN, J. B.; YINYU.; ZHANG, X. Solving large-scale sparse semidefinite programs for combinatorial optimization. **Siam Journal on Optimization**, v. 10, p. 443–461, 1998.

WOLSEY, L. A. **Integer programming**. [S.l.]: Wiley, 1998.

WU, Q.; HAO, J.-K. A memetic approach for the max-cut problem. In: SPRINGER. **International Conference on Parallel Problem Solving from Nature**. [S.l.], 2012. p. 297–306.